0-315-01058-4

National Library
of Canada

Canadian Theses Division

Ottawa, Canada
K1A 0N4

Bibliothèque nationale
du Canada

Division des thèses canadiennes

48916

PERMISSION TO MICROFILM — AUTORISATION DE MICROFILMER

- Please print or type — Écrire en lettres moulées ou dactylographier

Full Name of Author — Nom complet de l'auteur

Gerardus W.H. Coppus

Date of Birth — Date de naissance 2-7-1955	Country of Birth — Lieu de naissance Netherlands

Permanent Address — Résidence fixe

Hilversumstraat 11
Heerlen 6415 BD
Netherlands

Title of Thesis — Titre de la thèse

Robust Control of a Distillation Column

University — Université

University of Alberta

Degree for which thesis was presented — Grade pour lequel cette thèse fut présentée

Masters of Science

Year this degree conferred — Année d'obtention de ce grade 1980	Name of Supervisor — Nom du directeur de thèse Dr. S.L. Shah / Dr. R.K. Wood

Permission is hereby granted to the NATIONAL LIBRARY OF CANADA to microfilm this thesis and to lend or sell copies of the film.

The author reserves other publication rights, and neither the thesis nor extensive extracts from it may be printed or otherwise reproduced without the author's written permission.

L'autorisation est, par la présente, accordée à la BIBLIOTHÈQUE NATIONALE DU CANADA de microfilmer cette thèse et de prêter ou de vendre des exemplaires du film.

L'auteur se réserve les autres droits de publication; ni la thèse ni de longs extraits de celle-ci ne doivent être imprimés ou autrement reproduits sans l'autorisation écrite de l'auteur.

Date 16-10-1980	Signature

NL-91 (4/77)

National Library of Canada
Collections Development Branch

Canadian Theses on
Microfiche Service

Bibliothèque nationale du Canada
Direction du développement des collections

Service des thèses canadiennes
sur microfiche

NOTICE

The quality of this microfiche is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us a poor photocopy.

Previously copyrighted materials (journal articles, published tests, etc.) are not filmed.

Reproduction in full or in part of this film is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30. Please read the authorization forms which accompany this thesis.

THIS DISSERTATION
HAS BEEN MICROFILMED
EXACTLY AS RECEIVED

AVIS

La qualité de cette microfiche dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de mauvaise qualité.

Les documents qui font déjà l'objet d'un droit d'auteur (articles de revue, examens publiés, etc.) ne sont pas microfilmés.

La reproduction, même partielle, de ce microfilm est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30. Veuillez prendre connaissance des formules d'autorisation qui accompagnent cette thèse.

LA THÈSE A ÉTÉ
MICROFILMÉE TELLE QUE
NOUS L'AVONS REÇUE

Ottawa, Canada
K1A 0N4

NL-339 (Rev. 8/80)

THE UNIVERSITY OF ALBERTA

ROBUST CONTROL OF A DISTILLATION COLUMN

by

© G.W.M. Coppus B.Sc.

A THESIS
SUBMITTED TO THE FACULTY OF GRADUATE STUDIES AND RESEARCH
IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE
OF Master of Science
IN
Process Control

Chemical Engineering

EDMONTON, ALBERTA
Fall 1980

THE UNIVERSITY OF ALBERTA

FACULTY OF GRADUATE STUDIES AND RESEARCH

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies and Research, for acceptance, a thesis entitled ROBUST CONTROL OF A DISTILLATION COLUMN submitted by G.W.M. Coppus B.Sc. in partial fulfilment of the requirements for the degree of Master of Science in Process Control .

....

Supervisor

R. E. Rink..............................

D.G. Fisher..............................

Date....Sept 10 1980...................

Dedication

to Margo

Abstract

A robust multivariable controller design technique is outlined. The problems encountered when applying this technique to a nonlinear system are analyzed. To overcome these problems, it is proposed to apply the robust controller in a cascaded scheme with simple multiloop proportional or proportional-plus-derivative control in the slave loop. The performance of this control arrangement is evaluated on a nonlinear distillation column simulation model and compared to conventional PID control.

To apply the robust multivariable controller to a pilot plant distillation column, a control system utilizing an LSI-11 microcomputer has been developed. The system software and hardware, designed to provide flexibility and a comprehensive operator interface, are described in detail. The modifications necessary to adapt the system to different control algorithms are indicated.

The newly developed control system is used to apply multivariable robust control and multiloop PID control to a pilot plant distillation column. The robust controller is used both with proportional and proportional-plus-derivative control in the slave loop.

Comparison of the results shows that the robust controller generally performs better than conventional multiloop PID control.

Acknowledgement

The author would like to thank the Social Sciences and Humanities Research Council of Canada for financial support,
his supervisors Drs. Shah and Wood for their guidance and advice,
the technicians of the Instrument Shop and Machine Shop for their assistance in maintaining the equipment throughout this study,
his fellow students for their advice and help,
his wife for keeping the author in good spirits.

Table of Contents

Chapter Page

1. Introduction 1

2. Theory of Robust Control 4

  2.1 Introduction 4

  2.2 Theory of Robust Control 5

    2.2.1 Davison's Robust Feedforward - Robust Controller 5

    2.2.2 Feedforward - Robust Controllers and Frequency Domain Techniques 11

  2.3 Application of Robust Controller to a Nonlinear System 17

    2.3.1 Introduction 17

    2.3.2 Distillation Column, General 18

    2.3.3 Distillation Column, Numerical Example 22

  2.4 Modifications for a Nonlinear System 24

  2.5 Evaluation of Robust Controller on a Simulation Program 28

    2.5.1 Introduction 28

    2.5.2 The Distillation Column Simulation Program ..28

    2.5.3 Controller Design 33

    2.5.4 Robust Feedback-Feedforward Control 36

    2.5.5 Comparison to PI controllers 37

    2.5.6 Addition of Derivative Action 41

    2.5.7 Conclusions 42

3. Development of a Control System 44

  3.1 Introduction 44

  3.2 Requirements for the Control System 45

3.3 Hardware Configuration of Control System 46

3.4 Overall Data Acquisition and Control System 49

3.5 Development of the Control Software 52

3.5.1 Introduction 52

3.5.2 Buffer Usage in Foreground and Background Programs 55

3.5.3 The Foreground Program 59

3.5.4 The Background Program 60

3.5.5 Data Acquisition on Floppy Disks 62

3.5.6 Operator Commands 64

3.5.6.1 Run the foreground program: 65

3.5.6.2 Run the background program 65

3.5.6.3 The LIST command 66

3.5.6.4 The CHANGE command 67

3.5.6.5 Feed flow rate changes 70

3.5.6.6 Event scheduling 71

3.5.7 Modifying the Programs 73

3.6 Evaluation of the Control System 76

4. Experimental Robust Control of the Distillation Column 77

4.1 Introduction 77

4.2 Distillation Column Control 78

4.3 Distillation Column Equipment 81

4.4 Experiments for Controller Design 82

4.5 Robust Feedback Feedforward Control 88

4.6 Comparison of Robust and PID controllers 89

4.7 Robust Control plus Derivative Action 95

5. Conclusions 99

5.1 Controller Evaluation 99

5.2 Recommendations for future work 103

References 105

6. APPENDIX A: Description of programs and subroutines ..110

7. APPENDIX B: Plots of simulation and experimental responses 148

List of Tables

Table Page

1. Tuning of robust feedback controller..................38

2. Tuning of robust feedback feedforward controller......38

3. Robust feedback control vs. PI control................39

4. Robust feedback feedforward vs. PI+feedforward control40

5. PD-Robust feedback feedforward vs. PID + feedforward control...42

6. Tuning of robust feedback feedforward controller......90

7. Load disturbances and set point changes for robust feedback feedforward control..........................90

8. Load disturbances and set point changes for robust feedback control......................................90

9. Load disturbances and set point changes for robust feedback control vs. PI and PID control...............93

10. Load disturbances and set point changes for robust feedback feedforward control vs. PI and PID control..94

11. D-tuning for -25% feed flow disturbance..............96

12. P- and PD-robust feedback control vs. PID control.....97

13. P- and PD-robust feedback feedforward control vs. PID-plus-feedforward control..........................97

List of Figures

Figure Page

1. Plant representation in s-domain....................12
2. Block diagram of controlled plant....................14
3. Modified plant G(s)....................15
4. Plant with controller for set point tracking..........16
5. P-robust controlled plant....................25
6. Robust controller with combined master and slave loops29
7. Robust controller for simulation program..............31
8. Plant with slave control loop....................36
9. Distillation column with instrumentation for top loop.48
10. Schematic diagram of the distributed network of mini and micro computers and the interface with the computer controlled distillation column....................50
11. Overall flow diagram for foreground and background programs....................54
12. Flow diagram for foreground program..................61
13. Flow diagram for background program..................63
14. Pilot plant distillation column with instrumentation..83

List of Plots

Figure Page

SIMULATION RUNS

B1. Open loop, +5% steam flow rate........................149
B2. Open loop, -5% steam flow rate........................150
B3. P-control, +5% feed flow rate.........................151
B4. P-control, -5% feed flow rate.........................152
B5. P-robust feedback control, -20% feed flow rate......153
B6. PI-control, -20% feed flow rate......................154
B7. P-robust feedback control, +1% top product composition. 155
B8. PI-control, +1% top product composition..............156
B9. P-robust feedback control, +1% bottom product composition.....................................157
B10. PI-control, +1% bottom product composition..........158
B11. P-robust feedback feedforward control, -20% feed flow rate.......................................159
B12. PI-plus-feedforward control, -20% feed flow rate....160
B13. PD-robust feedback feedforward control, -20% feed flow rate.......................................161
B14. PID-plus-feedforward control, -20% feed flow rate...162
B15. PD-robust feedback feedforward control, +1% top product

composition..163
B16. PID-plus-feedforward control, +1% top product composition..164
B17. PD-robust feedback feedforward control, +1% bottom product composition..165
B18. PID-plus-feedforward control, +1% bottom product composition..166

EXPERIMENTAL RUNS

B19. Open loop, +10% reflux flow rate..167
B20. Open loop, -10% reflux flow rate..168
B21. P control (gains 3.571 top and -0.576 bottom), +20% feed flow rate..169
B22. P control (gains 2.5 top and -0.4 bottom), +10% feed flow rate..170
B23. P control (gains 2.5 top and -0.4 bottom), -10% feed flow rate..171
B24. P control (gains 1.786 top and -0.288 bottom), +10% feed flow rate..172
B25. P-robust feedback feedforward control, -25% feed flow rate..173
B26. P-robust feedback feedforward control, +25% feed flow rate..174
B27. P-robust feedback feedforward control, +1% top product composition..175
B28. P-robust feedback feedforward control, -1% top product

composition...176
B29. P-robust feedback feedforward control, +1% bottom product composition...177
B30. P-robust feedback feedforward control, -1% bottom product composition...178
B31. PI control, -25% feed flow rate...179
B32. PID control, -25% feed flow rate...180
B33. PID-plus-feedforward control, -25% feed flow rate...181
B34. PID-plus-feedforward control, +1% top product composition...182
B35. PID-plus-feedforward control, +1% bottom product composition...183
B36. PD-robust feedback feedforward control, +25% feed flow rate...184
B37. PD-robust feedback feedforward control, -25% feed flow rate...185
B38. PD-robust feedback feedforward control, +1% top product composition...186
B39. PD-robust feedback feedforward control, -1% top product composition...187
B40. PD-robust feedback feedforward control, +1% bottom product composition...188
B41. PD-robust feedback feedforward control, -1% bottom product composition...189

Nomenclature

Symbol Meaning

x = state vector

A = system matrix

B = input matrix

C = output matrix

u = control input

E = disturbance matrix

w = measurable disturbance vector

e = error vector

y = output vector

D = direct input matrix

K_P = proportional gain

K_I = integral gain

K_D = derivative gain

T_I = integral time

T_D = derivative time

T_{d_1} = derivative time for one loop

K_{d_1} = derivative gain for one loop

k_1 = proportional gain for one loop

T_1' = steady state tracking gain matrix

D = steady state disturbance gain matrix

ε = tuning constant for robust control

1. Introduction

There is a considerable interest in the design of controllers for multivariable systems [2,3]. Most of the modern control synthesis techniques require a mathematical model of the process that is to be controlled. In many instances the modelling of complex, often poorly understood processes is a substantial and crucial task and by no means routine. A promising alternative for the control of such processes is to employ design methods that require only plant input-output data and not an explicit mathematical model of the process. Self-tuning methods that have arisen largely out of the work of Astrom and co-workers [4-6] have been used successfully to control a number of industrial processes. A more recent technique introduced by Davison [1] relies only on plant input-output data and is based on the notion of compensator identification as opposed to plant identification [7-10]. It is to be noted that Davison's controller and Francis and Wonham's [11] controller based on the internal model principle are very similar in structure in that both require an internal model. A good comparison of the two methodologies is described by Kwatny and Kalnitsky [12].

Two limitations apply to Davison's technique: the plant characteristics should be such that it can be described by a linear, time-invariant model; and it is also required to be open-loop asymptotically stable. However, unlike the self tuning methods, Davison's robust feedback-feedforward technique does not require any assumptions regarding the plant model order , minimum phase characteristics, etc. Secondly, the technique guarantees a robust (with respect to small parameter variations), closed loop, asymptotically stable, multivariable system. In contrast, convergence and stability properties of even single-loop systems under self-tuning control are difficult to analyze or guarantee [13].

The availability of sophisticated computing hardware and the development of complex control strategies have triggered a rapid development of computer control systems [19,20,22,23]. The tendency is away from single maxi computers to hierarchial networks of mini computers as these provide more flexibility and reliability combined with more than enough computing power. The detailed development of one "node" in a control network is necessary for the implementation of complex process control algorithms.

The purpose of this thesis is twofold: the evaluation of Davison's robust control technique on a realistic plant and the development of the tools to do so.

This thesis is organized as follows. In chapter 2 the theory of robust multivariable control is outlined. The difficulties posed by nonlinear systems are indicated and a simple but elegant way of solving those problems is proposed. The chapter is concluded with an extensive evaluation of the controller on a simulation program.

Chapter 3 describes the development of a control system which involves putting together the necessary hardware and developing the software. Although the system was developed with the department's pilot plant distillation column in mind, it can easily be adapted to other systems.

Chapter 4 is the synthesis of chapters 2 and 3: the application of the robust control technique on the distillation column. An extensive series of experiments forms the basis for the controller evaluation.

The last chapter contains the conclusions and recommendations for future work.

2. Theory of Robust Control

2.1 Introduction

A majority of the advanced control techniques deal with idealized linear systems and are generally more complex than conventional PID control and require in depth knowledge of the plant that is to be controlled. In comparison to this, the design and application of a multivariable tuning technique due to Davison is simple and straightforward. However, like most other methods it has been developed for idealized linear systems, its performance for nonlinear plants is unknown and a subject of interest.

At this point we should point out that there seems to be no single definition of the term "robustness" in the literature. Generally robustness of a control system indicates the fact that the system performance is maintained in the face of uncertainties or differences between plant model and actual plant. Usually when we think of robustness, we specify gain and phase margins for single loop linear systems.

A possible measure of robustness for a feedback system is the magnitude of (otherwise arbitrary) perturbations which may be tolerated without instability. By "perturbations" is meant the deviation of the plant transfer function model G(s) from the true plant [14].

Davison [1] does not quantify robustness, but proves that asymptotic tracking and output regulation occurs independent of input disturbances and plant parameter variations, as long as the plant is stable and is describable by a finite dimensional, linear, time-invariant state-space model.

In this chapter we will outline the theory as originally published by Davison [1]. Since the theory was developed for linear systems, we may expect to encounter some problems when applying the Robust Controller directly to a real life nonlinear system. We will look at the character of the difficulties and its causes.

Then we will make a proposal to deal with the nonlinear system to make its behaviour acceptable for the robust controller by incorporating slave control loops in the total control system. Several advantages to this approach will be outlined.

Finally, we will design and evaluate a Robust Controller for a nonlinear distillation column simulation program.

2.2 Theory of Robust Control

2.2.1 Davison's Robust Feedforward - Robust Controller

Consider the following linear time-invariant stable plant:

$$\dot{x} = Ax + Bu + Ew$$

$$e = Cx + Du + Fw - y$$

$$y(s) = C(sI-A)^{-1}Bu(s) + C(sI-A)^{-1}Ew(s) + Du(s) \qquad (1)$$

where $x \in R^n$ is the state, $u \in R^m$ is the control input, $e \in R^r$ is the error which is the difference between the output

$$y = Cx + Du + Fw$$

and the specified reference input y_{ref}, where $y_{ref} \in R^r$ and $w \in R^d$ is an input disturbance.

It is assumed that the elements of w and y_{ref} satisfy the same differential equation

$$\dot{w}^{(p)} + a_p \dot{w}^{(p-1)} + \ldots\ldots + a_2 \dot{w} + a_1 w = 0 \tag{2}$$

which has characteristic roots all contained in the closed right hand part of the complex plane and that the initial conditions of (2) are independent of each other for each element of w and y_{ref}.

We will only consider step changes in set points and disturbance inputs. For this type of inputs (2) becomes

$$\dot{w} = 0 \tag{3}$$

with characteristic root $\lambda_1 = 0$ which is contained in the closed right hand part of the complex plane.

From Davison's more general theorems we can derive directly:

Theorem 1

If the disturbance w is measurable then there exists a feed forward controller for the system (1) so that $e(t) = (y_{ref} - y) \rightarrow 0$ as $t \rightarrow \infty$ for all constant disturbances $w \in R^d$ and constant reference inputs $y_{ref} \in R^r$ if and only if

$$\operatorname{rank} G(0) = \operatorname{rank} C(sI-A)^{-1}B + D\big|_{s=0} = r \tag{4}$$

Theorem 2

There exists a robust controller for the system (1) such that $e(t) = (y_{ref} - y) \rightarrow 0$ as $t \rightarrow \infty$ for all constant disturbances $w \in R^d$ and all constant reference inputs $y_{ref} \in R^r$ and the system is asymptotically stable if and only if

$$\operatorname{rank} G(0) = \operatorname{rank} C(sI-A)^{-1}B + D\big|_{s=0} = r \tag{5}$$

Davison's T_1^1-matrix is equivalent to $G(0)$ used in frequency domain techniques.

The controller identification procedure

If the conditions of theorems 1 and 2 are satisfied, we can proceed with the controller identification. We will consider a controller for tracking of constant set points and regulatory control under constant disturbances, i.e. the controller will be designed for step changes in set points and disturbances, not for ramps, parabolas, etc.

We have to perform two sets of experiments on the plant to find the Steady-State Tracking Gain Matrix and the Steady-State Disturbance Gain Matrix:

Experiment I (to find T_1^1)

Step 1: Apply an input $u(t) = \overline{u}_1$, where $\overline{u}_1 \in R^m$, $\overline{u}_1 \neq 0$; then since the system (1) is stable, there exists a constant vector $\overline{y}_1'$ so that

$$\lim_{t\to\infty} \{y(t) - \overline{y}_1'\} = 0$$

Measure $\overline{y}_1'$.

In other words, measure the steady-state response $\overline{y}_1'$ to a step input $\overline{u}_1$.

Step 2: Apply an input $u(t) = \overline{u}_2$, where $\overline{u}_2 \in R^m$ and $\overline{u}_2$ is linearly independent of $\overline{u}_1, \overline{u}_3, \ldots, \overline{u}_m$; then there exists a constant vector $\overline{y}_2'$ so that

$$\lim_{t\to\infty} \{y(t) - \overline{y}_2'\} = 0$$

Measure $\overline{y}_2'$.

In other words, measure the steady-state response $\overline{y}_2'$ to a step input $\overline{u}_2$ which is independent of all other step inputs in experiment I.

.

.

.

.

Step m: Apply an input $u(t) = \overline{u}_m$, where $\overline{u}_m \in R^m$ and $\overline{u}_m$ is linearly independent of $\overline{u}_1, \overline{u}_2, \ldots, \overline{u}_{m-1}$; then there exists a constant vector $\overline{y}_m'$ so that

$$\lim_{t\to\infty} \{y(t) - \overline{y}_m^1\} = 0$$

Measure $\overline{y}_m^1$.

Then,

$$T_1^1 = (\overline{y}_1^1, \overline{y}_2^1, \ldots, \overline{y}_m^1)(\overline{u}_1, \overline{u}_2, \ldots, \overline{u}_m)^{-1}$$

Experiment II (to find D_1^1)

It is now assumed that the disturbance inputs w can be excited. We perform exactly the same procedure as in Experiment I, but this time we excite the disturbance inputs instead of the control inputs:

Step 1: Apply an input $w(t) = \overline{w}_1$, where $\overline{w}_1 \in R^d$, $\overline{w}_1 \neq 0$; then we find a constant vector $\overline{y}_1^d$ so that

$$\lim_{t\to\infty} \{y(t) - \overline{y}_1^d\} = 0$$

Measure $\overline{y}_1^d$.

.

.

.

Step d: Apply an input $w(t) = \overline{w}_d$, where $\overline{w}_d \in R^d$ and $\overline{w}_d$ is linearly independent of $\overline{w}_1, \overline{w}_2, \ldots, \overline{w}_{d-1}$.

Then we find a constant vector $\overline{y}_d^d$ so that

$$\lim_{t\to\infty} \{y(t) - \overline{y}_d^d\} = 0$$

Measure $\bar{y}_d^d$.

Then

$$D_1^1 = (\bar{y}_1^d, \bar{y}_2^d, \ldots, \bar{y}_d^d)(\bar{w}_1, \bar{w}_2, \ldots, \bar{w}_d)^{-1}$$

and the feedforward - Robust controller is given by:

$$u = [T_1^1]^+ y_{ref} - [T_1^1]^+ D_1^1 w - [T_1^1]^+ \varepsilon \int_0^t (y - y_{ref}) dt' \tag{6}$$

where T_1^1 = tracking steady state gain matrix

D_1^1 = disturbance steady state gain matrix

y_{ref} =set points

y = plant outputs

w = disturbance inputs

ε =tuning constant

+ denotes pseudo inverse:

$[T_1^1]^+ = [T_1^1]' [T_1^1 T_1^{1\prime}]^{-1}$

for a square matrix $[T_1^1]^+ = [T_1^1]^{-1}$

2.2.2 Feedforward - Robust Controllers and Frequency Domain Techniques

In the previous section we looked at the feedforward - robust controller in the time domain. In this section we try to translate some results of (state-space) robust controller design to frequency domain design.

The feedforward - robust controller for a stable plant and constant disturbances was given by (6),

$$u = [T_1^1]^+ y_{ref} - [T_1^1]^+ D_1^1 w - [T_1^1]^+ \varepsilon \int_0^t (y - y_{ref})\, dt' \qquad (6)$$

As we saw, T_1^1 and D_1^1 can be determined by some simple steady state experiments. Since we are interested in frequency domain techniques, we consider the relationship between T_1^1, D_1^1 and the transfer function matrices. Figure 1 shows the plant configuration.

Column 1 of T_1^1 contains the steady state response of the output vector y to a unit step in u_1, column 2 the response of y to a unit step in u_2, etc. Hence it is obvious that T_1^1 is the same as G(0) and in the same way $D_1^1 = D(0)$.

Generally,

$$T_k^t = (-1)^{t-1} \frac{1}{(t-1)!} \frac{d^{t-1}}{ds^{t-1}} G(s) \Big|_{s=\lambda_k} \qquad (7)$$

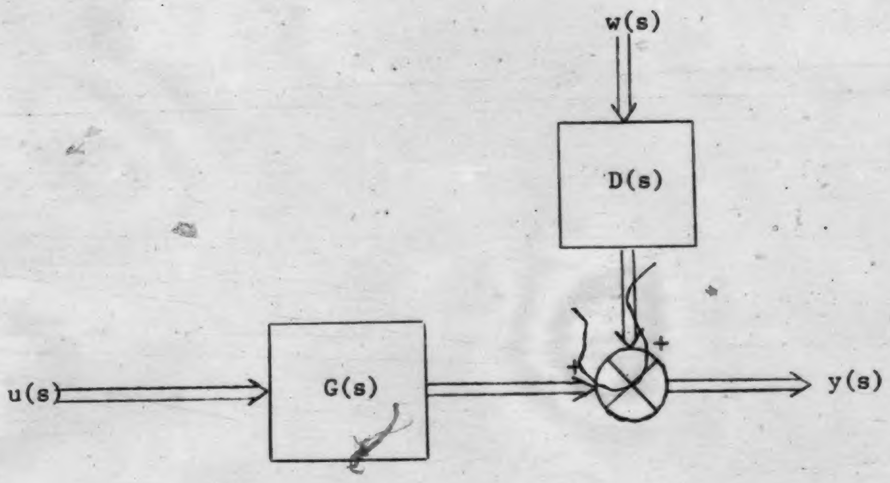


Figure 1. Plant representation in s-domain

where

$t = 1, 2, \ldots, q_k$, q_k is the multiplicity of root

$k = 1, 2, \ldots, \bar{p}$, $\bar{p}$ is the number of different characteristic roots of (2)

The same type of relationship exists for D_1', replace G(s) by D(s) in (7).

We consider constant inputs and disturbances, so equation (3), $\dot{w} = 0$, applies. This has one root, $\lambda_1 = 0$ and from (7) we find,

$$T_1' = G(s)\Big|_{s=\lambda_1=0} = G(0) \qquad (8)$$

We can rewrite (6) as

$$u(s) = [G(0)]^{+}\{y_{ref}(s) - D(0)w(s) - \frac{\varepsilon}{s}(y(s)-y_{ref}(s)\} \quad (9)$$

See figure 2 for the block diagram.

The closed loop transfer function is

$$\begin{aligned} y(s) &= G(s)u(s) + D(s)w(s) \\ &= G(s)G(0)^{+}\{y_{ref}(s) - D(0)w(s) - \frac{\varepsilon}{s}(y(s) \\ &\quad - y_{ref}(s))\} + D(s)w(s) \qquad (10) \end{aligned}$$

$$\begin{aligned} y(s) &= [I + \frac{\varepsilon}{s}G(s)G(0)^{+}]^{-1}G(s)G(0)^{+}(1 + \frac{\varepsilon}{s})y_{ref}(s) \\ &\quad + [I + \frac{\varepsilon}{s}G(s)G(0)^{+}]^{-1}(D(s) - G(s)G(0)^{+}D(0))w(s) \quad (11) \end{aligned}$$

When we apply step inputs to y_{ref} and w, the final value theorem shows

$$\lim_{t\to\infty} y(t) = \lim_{s\to 0} sy(s) = \bar{y}_{ref} \qquad (12)$$

A possible design procedure for a constant set point and disturbances controller is:

1. Using DNA (Direct Nyquist Array) and Characteristic

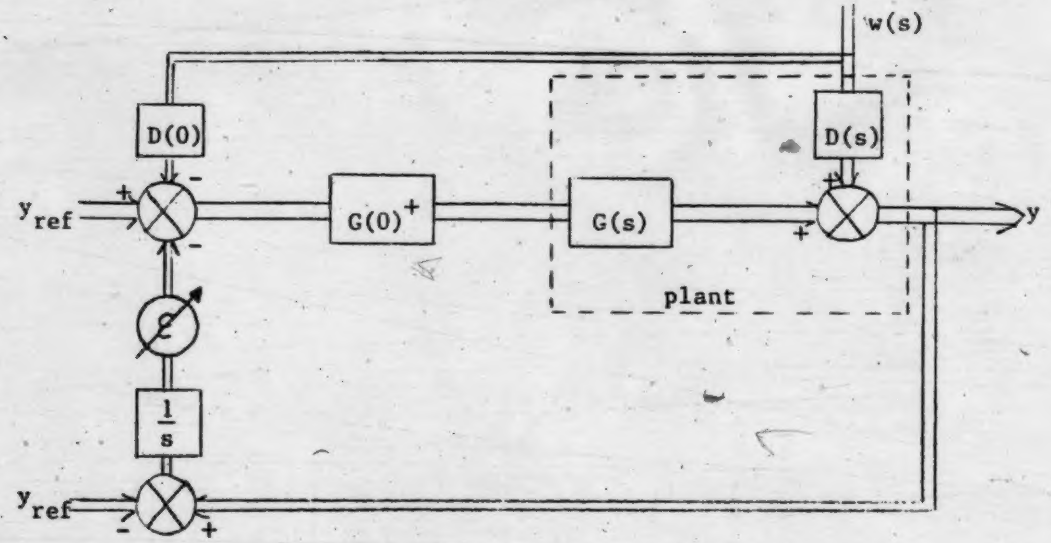


Figure 2. Block diagram of controlled plant

Locus Techniques, design a controller K(s) for the plant Q(s), such that the resultant closed loop system G(s) is stable (see fig. 3) [15,16]

2. Using DNA, obtain G(0).
3. Implement the control scheme as shown in figure 2.
4. Tune the controller by varying epsilon.

We will briefly look at a feedforward controller for constant disturbances and tracking set points. In this case the set points satisfy the differential equation

$$\ddot{y}_{ref} = 0 \tag{13}$$

which has characteristic root $\lambda_1 = 0$ with multiplicity $q_1 = 2$.

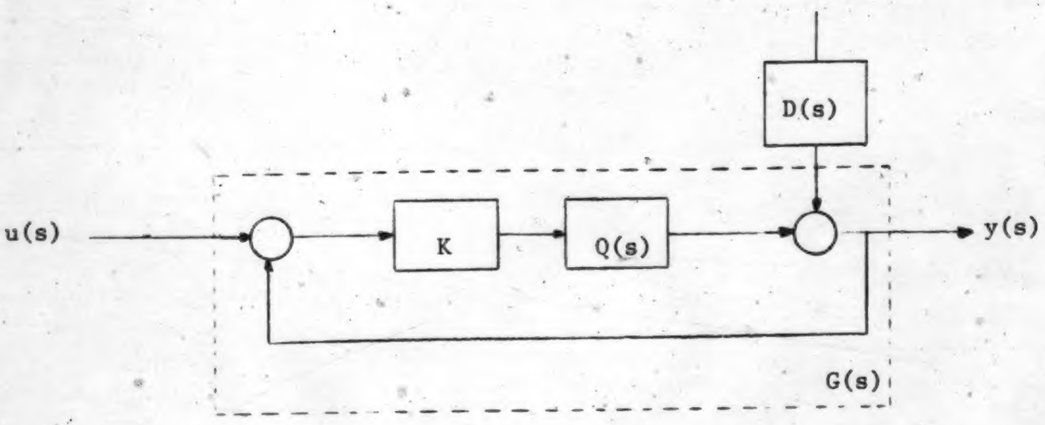


Figure 3. Modified Plant G(s)

For the controller we need

$$T_1^1 = G(0) \tag{14}$$

$$T_1^2 = -\left.\frac{dG(s)}{ds}\right|_{s=0} = -G'(0) \tag{15}$$

and the controller is given by

$$\begin{aligned} u(s) = &- G(0)^+ D(0) w(s) + G(0)^+ y_{ref}(s) \\ &- G(0)^+ G'(0) G(0)\, s y_{ref}(s) \\ &- G(0)^+ \left(\frac{\varepsilon_1}{s} + \frac{\varepsilon_2}{s^2}\right)(y(s) - y_{ref}(s)) \end{aligned} \tag{16}$$

See figure 4 for the block diagram

We have seen that the robust controller design technique can be useful in frequency domain design. However

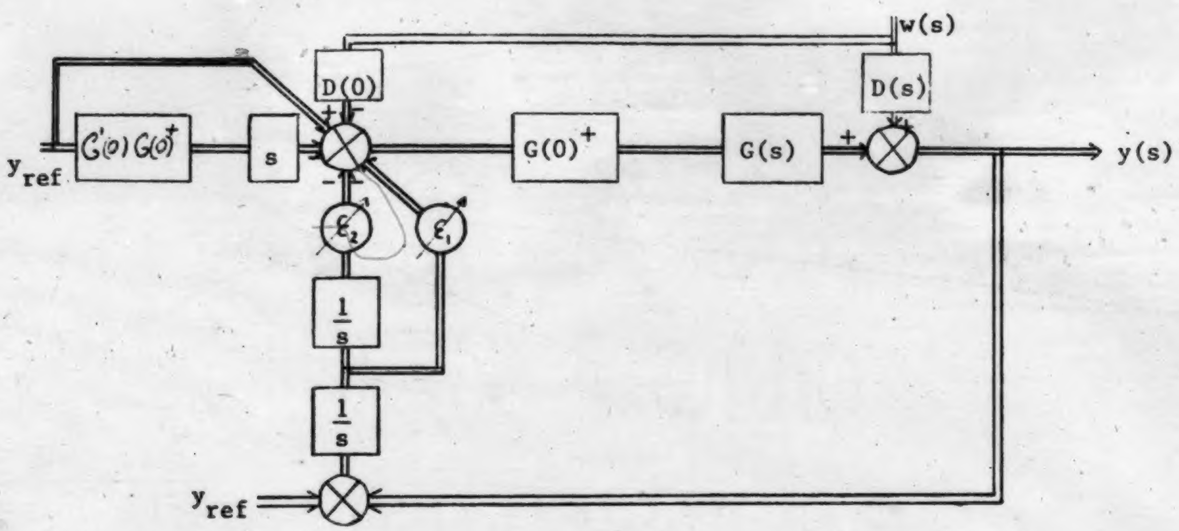

w(s)
D(0)
D(s)
$G'(0)G(0)^+$
s
$G(0)^+$
G(s)
y(s)
y_{ref}
+
−
+
ε_2
ε_1
$\frac{1}{s}$
$\frac{1}{s}$
y_{ref}

Figure 4. Plant with controller for set point tracking

if the desired outputs or disturbances follow higher order differential equations, the design becomes more cumbersome because we have to evaluate derivatives of G(s) and $G(\lambda_k)$. This is not easily done by graphical methods since we usually work with $G(\omega)$.

2.3 Application of Robust Controller to a Nonlinear System

2.3.1 Introduction

In the following discussion, it will be assumed that the system under consideration is 2x2, i.e. two inputs and two outputs. For a square matrix $[T_1']^+ = [T_1']^{-1}$.

As we outlined in the previous chapters, we use steady state data for the design of a feedback controller:

$$u = -[T_1']^{-1} \varepsilon \int_0^t (y - y_{ref})dt' \qquad (17)$$

If we change the sign but not the magnitude of a control input, only the sign of the (steady-state) plant output will change, not the magnitude. When we design a robust feedback controller for a linear plant, both positive and negative control inputs will lead to the same controller as we will see.

In a nonlinear plant, the effects of positive and negative inputs will often cause outputs to be different in magnitude as well as sign. In that case, we will find two different feedback controllers for the same operating point of a nonlinear plant, one for positive inputs (disturbances) and one for negative inputs.

The most serious difference in the controllers would be a difference in sign, i.e. one controller would perform positive feedback and the other negative feedback. In that case the controllers would have to be rejected for stability reasons. We will investigate under what circumstances a sign change will occur. The effect of changes in magnitude of the entries in the feedback controller matrix will not be considered.

2.3.2 Distillation Column, General

Let the inputs and outputs of a distillation column be denoted by:

u_1 = reflux flowrate	$y_{1.}$ = top composition
u_2 = steam flowrate	$y_{2.}$ = bottom composition

All variables are perturbation variables

y_{21} = change in bottom composition resulting from a change in reflux flowrate.

In the controller, the most important role is played by the matrix

$$[T_1']^{-1} = \left\{ \begin{bmatrix} y_{11} & y_{12} \\ y_{21} & y_{22} \end{bmatrix} \begin{bmatrix} u_1 & 0 \\ 0 & u_2 \end{bmatrix}^{-1} \right\}^{-1}$$

$$= \begin{bmatrix} \frac{y_{11}}{u_1} & \frac{y_{12}}{u_2} \\ \frac{y_{21}}{u_1} & \frac{y_{22}}{u_2} \end{bmatrix}^{-1} \qquad (18)$$

write (18) as

$$[T_1^1]^{-1} = \begin{bmatrix} K_{11} & K_{12} \\ K_{21} & K_{22} \end{bmatrix}^{-1} \qquad (19)$$

where K's are steady-state gain parameters.

If u_1 and u_2 are positive, then y_{11} and y_{21} will be positive (i.e. an increase in reflux flowrate will cause an increase in both top and bottom composition), and y_{12} and y_{22} will be negative.

Denote the steady-state gain parameters for positive inputs with a superscript P:

K_{11}^P , K_{21}^P are positive and K_{12}^P , K_{22}^P are negative.

If u_1 and u_2 are negative, y_{11} and y_{21} will be negative and y_{12} and y_{22} will be positive, but

K_{11}^N , K_{21}^N are positive and K_{12}^N , K_{22}^N are negative .

Ergo, the entries in the steady-state gain matrix have the same signs for positive and negative inputs. But for control we use the *inverse* of the steady-state gain matrix.

The entries in the inverse of the K^P- matrix will have opposite signs of the entries in the K^N- matrix if

$$\text{sign}(\det K^P) \neq \text{sign}(\det K^N)$$

We will now investigate what the physical meaning of a sign change is.

Consider det K^P :

$$\det K^P = K_{11}^P K_{22}^P - K_{12}^P K_{21}^P \qquad (20)$$

From steady-state data, 10% change in inputs we find,

$$\det K^P > 0, \tag{21}$$

$$K^P_{11}\,K^P_{22} - K^P_{12}\,K^P_{21} > 0\ , \tag{22}$$

$$K^P_{11}\,K^P_{22} > K^P_{12}\,K^P_{21}\quad , \tag{23}$$

Taking the signs of K's into account, we can write (23) as

$$\frac{K^P_{11}}{K^P_{21}} < \frac{K^P_{12}}{K^P_{22}} \tag{24}$$

$$\text{or }, \ r^P_1 < r^P_2 \tag{25}$$

The parameter r is a gain ratio and can be considered as a measure of selectivity:

r^P_1 > 1 means control input 1 affects the top product composition stronger than the bottom product composition,

r^P_1 = 1 means control input 1 affects top and bottom product composition equally strong,

r^P_1 < 1 means control input 1 affects the bottom product composition stronger than the top product composition.

Expression (25) in words is:

for positive control inputs, the selectivity of the steam flowrate (input 2) for the top product composition is stronger than the selectivity of the reflux flowrate for the top product composition.

For negative inputs we find

$$\det K^N = K_{11}^N K_{22}^N - K_{12}^N K_{21}^N < 0 \; ; \qquad (26)$$

hence

$$\frac{K_{11}^N}{K_{21}^N} > \frac{K_{12}^N}{K_{22}^N} \qquad (27)$$

or , $r_1^N > r_2^N$ (28)

Expression (28) in words is:

for negative control inputs, the selectivity of the reflux flowrate for the top product composition is stronger than the selectivity of the steam flowrate for the top product composition.

We see that the emphasis in the response has changed, now the most effective way of influencing the top product composition with as little effect as possible on the bottom product composition appears to be a change in reflux flowrate.

Somewhere between the situations det K negative and det K positive we expect the case of det K = 0. For a linear system this would mean uncontrollability since the outputs would no longer be independent, i.e. y_1 would be a multiple of y_2 for every combination of inputs. However, in our nonlinear situation, det K is not independent of the sign and magnitude of the inputs and we cannot speak of "dependent outputs for every combination of inputs". Therefore, in this case the occurence of det K = 0 has only mathematical significance, indicating a point where Davison's synthesis technique is not applicable.

Simulation and experimental data elsewhere in this thesis will show that we indeed get a sign change in controllers designed with Davison's method if we use negative instead of positive inputs and disturbances in the plant experiments.

2.3.3 Distillation Column, Numerical Example

Some open loop experiments were performed on a nonlinear distillation column simulation program (see section 2.5.2) to illustrate the effect of nonlinearities. The inputs in this case were reflux flow rate and steam flow rate, the outputs were top product composition and bottom product composition.

It was found that for positive inputs:

$$[T_1^1]^{-1} = \left\{ \begin{bmatrix} 0.788 & -2.765 \\ 4.319 & -3.981 \end{bmatrix} \begin{bmatrix} 1.21 & 0 \\ 0 & 1.458 \end{bmatrix}^{-1} \right\}^{-1}$$

$$= \begin{bmatrix} 0.651 & -1.896 \\ 3.569 & -2.731 \end{bmatrix}^{-1} = [K^P]^{-1}$$

$$\det K^P = 4.99$$

$$r_1^P = 0.182$$

$$r_2^P = 0.823$$

$$\rightarrow \quad r_1^P < r_2^P$$

$$[T_1^1]^{-1} = \begin{bmatrix} -0.547 & 0.380 \\ -0.715 & 0.131 \end{bmatrix}$$

and for negative inputs:

$$[T_1^1]^{-1} = \left\{ \begin{bmatrix} -1.291 & 0.991 \\ -2.335 & 12.456 \end{bmatrix} \begin{bmatrix} -1.21 & 0 \\ 0 & -1.458 \end{bmatrix}^{-1} \right\}^{-1}$$

$$= \begin{bmatrix} 1.067 & -0.680 \\ 1.930 & -8.543 \end{bmatrix}^{-1} = [K^N]^{-1}$$

$\det K^N = -7.80$

$r_1^N = 0.553$

$r_2^N = 0.080$

$\rightarrow \quad r_1^N > r_2^N$

$$[T_1^1]^{-1} = \begin{bmatrix} 1.095 & -0.087 \\ 0.247 & -0.137 \end{bmatrix}$$

As we can see, the signs in the $[T_1^1]^{-1}$ matrices are opposite, which means that at least one of the matrices would produce an unstable controller (positive feedback). Also the magnitude of the entries is very different in both matrices. We may conclude that the robust controller design technique simply does not work with this plant.

2.4 Modifications for a Nonlinear System

As we saw in previous sections, the nonlinear behaviour of plants like the distillation column makes direct application of the robust controller impossible because the controller design gives ambiguous results.

We suggest a rather simple but practically useful modification to extend the applicability of the robust design technique to a larger class of processes.

Utilization of simple multiloop proportional or proportional + derivative control for a typical plant achieves two very important plant characteristics. With the correct choice of gains the plant can be made asymptotically stable and secondly the assumption of linearity is valid over a wider range of operating conditions than the original plant, due to the effects of K in the forward path and the unity gain feedback loop. The robust controller can be synthesized in such a cascaded scheme and implemented as a master loop as shown in figure 5. The slave or inner loop consists of the original plant with multiloop proportional feedback gain K, or proportional plus derivative feedback. It is not possible to have multiloop PI control in the inner or slave loop because this would eliminate steady-state offset in the outputs and result in $[T_1'] = \underline{0}$, suggesting non-existence of the robust controller [1].

The above suggested cascaded scheme has three advantages. Firstly, the proportional feedback action may make the plant appear more linear, i.e., the excursions due

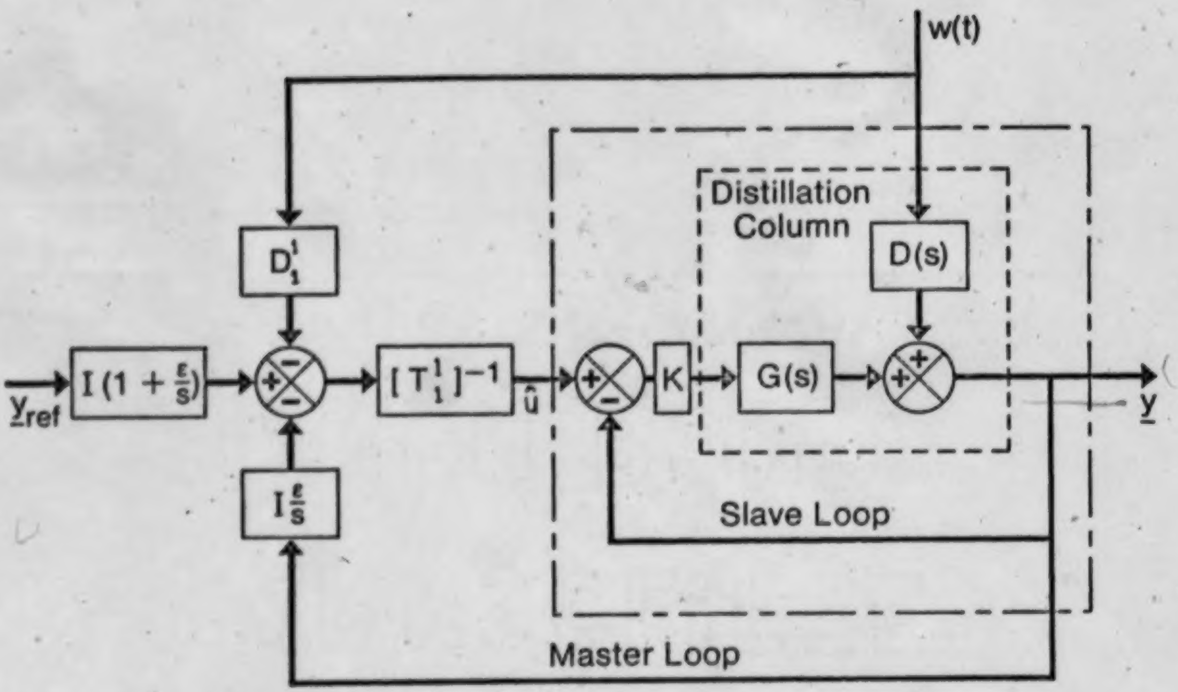

w(t)
Distillation Column
D(s)
D_1^1
$I(1+\frac{\varepsilon}{s})$
$[T_1^1]^{-1}$
$\hat{u}$
K
G(s)
$\underline{y}_{ref}$
$\underline{y}$
$I\frac{\varepsilon}{s}$
Slave Loop
Master Loop
+
−

Figure 5. P-Robust controlled plant

to set point interactions and disturbances are reduced. The linearity assumption can now be considered to be valid and Theorems I and II of Davison and the resulting algorithms can be applied to the augmented 'linear' plant. Secondly, in the suggested configuration it is possible to design the robust controller for an industrial unit without having to do experiments I and II on the open-loop plant. The normal steady-state input-output data would be sufficient for the controller design. Thirdly, the plant inherently has analog backup in the form of proportional (or PD) feedback control in the event of a computer failure. The two latter features are very important for field implementation of advanced control strategies on industrial plant units.

For a plant described by eqn. (1), the augmented plant with controller, $u = K\hat{u} - Ky$, becomes

$$\begin{aligned} \dot{x} &= (A - BKC)x + BK\hat{u} + Ew \\ y &= Cx \end{aligned} \tag{29}$$

$\hat{u}$ is the set point for the slave loops

Thus the modified plant is characterized by matrices $(A-BKC) = \hat{A}$, $BK = \hat{B}$, and $E = \hat{E}$.

In case of PD slave control, the augmented plant will be slightly different. The control law is:

$$\begin{aligned} u &= -K_p y - K_D\dot{y} + K_p\hat{u} \\ &= -K_p Cx - K_D C\dot{x} + K_p\hat{u} \end{aligned} \tag{30}$$

where K_p is the proportional gain in the slave loop and K_D is the derivative gain.

Then,

$$\begin{aligned} \dot{x} &= Ax - BK_p Cx - BK_D C\dot{x} + BK_p\hat{u} + Ew \\ (I + BK_D C)\dot{x} &= (A - BK_p C)x + BK_p\hat{u} + Ew \\ \dot{x} &= [I + BK_D C]^{-1}(A - BK_p C)x + [I + BK_D C]^{-1}BK_p\hat{u} \end{aligned} \tag{31}$$

$$+ [I + BK_D C]^{-1} Ew \tag{32}$$

Rewrite equation (32) as:

$$\dot{x} = \hat{A}x + \hat{B}\hat{u} + \hat{E}w \tag{33}$$

To guarantee existence of a robust controller for the modified system, Davison's results in terms of $\hat{A}$, $\hat{B}$ and $\hat{E}$ can be restated as follows:

Theorem 1'

If the disturbance w is measurable then there exists a feedforward controller for *the augmented system* (29,33) so that $e(t) = (y_{ref} - y) \to 0$ as $t \to \infty$ for all constant disturbances $w \in R^d$, and constant reference inputs $y_{ref} \in R^r$ if and only if

$$\text{rank } G'(0) = \text{rank } C(sI-\hat{A})^{-1} \hat{B}\Big|_{s=0} = r$$

Theorem 2'

There exists a robust controller for *the augmented system (29,33)* such that $e(t) \to 0$ as $t \to \infty$ for all constant disturbances $w \in R^d$ and all constant reference inputs $y_{ref} \in R^r$ and the closed loop system is asymptotically stable if and only if

$$\text{rank } G'(0) = \text{rank } C(sI-\hat{A})^{-1} \hat{B}\Big|_{s=0} = r$$

Theorems 1' and 2' are elementary extensions of Davison's results [1].

Having ascertained the existence of a robust feedback, feedforward controller, the final controller is given by:

$$u = [T_1']^{-1} y_{ref} - [T_1']^{-1} D_1' w - [T_1']^{-1} \varepsilon \int_0^t (y - y_{ref}) dt' \qquad (34)$$

where $T_1' = G'(0)$ and $D_1' = C(sI - \hat{A})^{-1} \hat{E}$ and $\varepsilon > 0$.

It is interesting to note that block diagram manipulation of this scheme shows that the overall 'robust' control scheme is in fact a multivariable PI scheme as can be seen from figure 6. Notice also the different amount of proportional action taken on the y_{ref} and y signals. From a frequency domain viewpoint the controller matrix $[T_1']^{-1}$ is simply a steady-state decoupling matrix.

2.5 Evaluation of Robust Controller on a Simulation Program

2.5.1 Introduction

In this chapter we will describe the design and implementation of a robust feedback feedforward controller on a nonlinear distillation column simulation program.

First we will have a look at the program itself and the way the slave loop control was implemented.

Then we will design a robust controller for the simulated plant, and finally we will tune the robust controller and compare its performance to a conventional PID controller.

2.5.2 The Distillation Column Simulation Program

The simulation package that was used in this study is an extension of the original programs that were developed by Svrcek [17] to model the department's pilot distillation

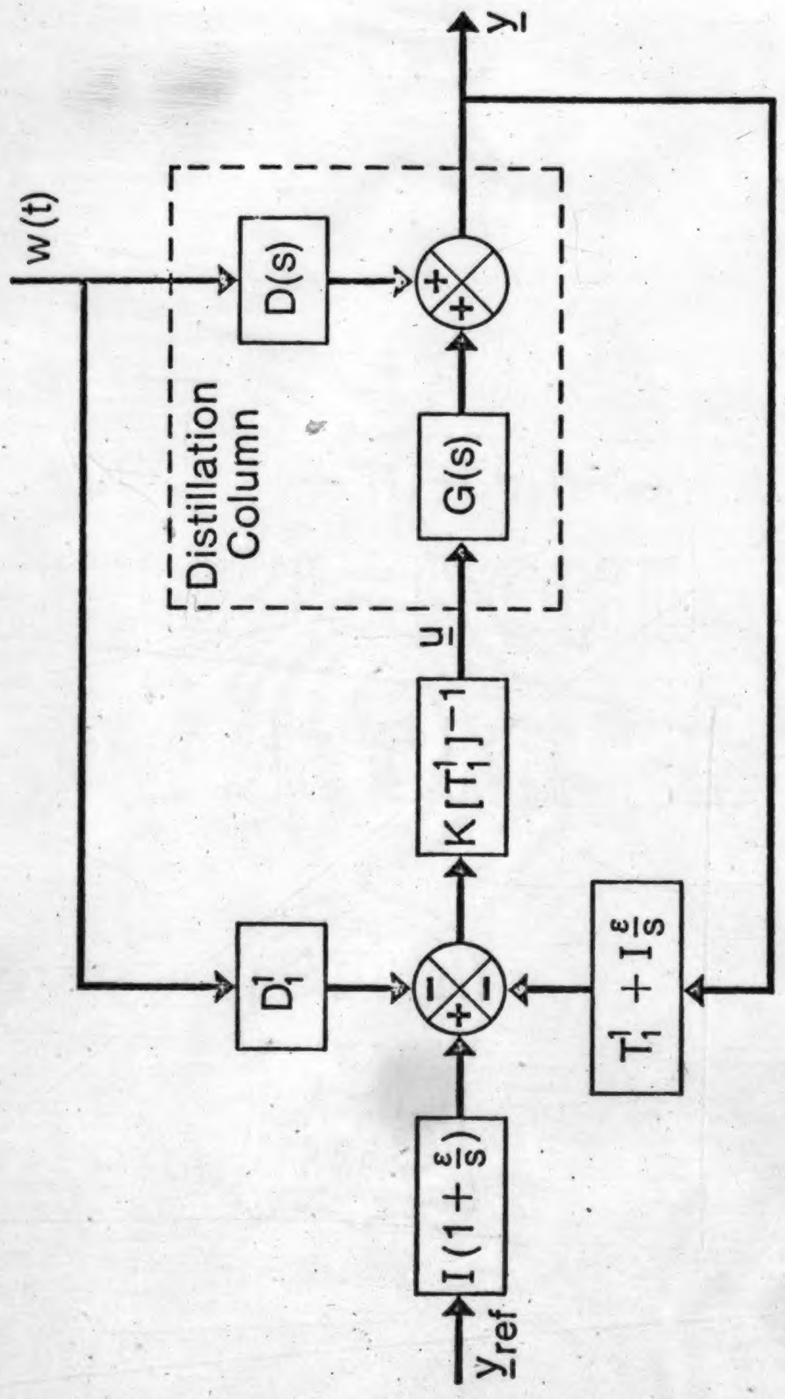

w(t)
D(s)
Distillation Column
G(s)
y
u
K[T$_1^{-1}$]$^{-1}$
D$_1^1$
T$_1^1$ + I $\frac{\varepsilon}{s}$
I (1 + $\frac{\varepsilon}{s}$)
y$_{ref}$
+
-

Figure 6. Robust controller with combined master and slave loops

column.

The model is nonlinear and based on material and energy balances. Because the model presents an adequate representation of binary distillation column behaviour, it is the most useful tool for evaluating controller behaviour short of actual plant implementation. Due to changes in the equipment and operating conditions of the pilot scale distillation column (described in part 4 of this thesis), which are not yet reflected by the simulation programs, we do not claim that the model accurately presents the dynamics of the pilot plant distillation column. Therefore the simulations and experimental evaluation should be seen as two separate entities.

In figure 7 we see the proposed implementation of the robust controller. Note that the proportional controller in the slave loop is situated in the feedback loop and not in the forward path as originally proposed in section 2.4. The reasons for this are:

1. On a real plant one could have a conventional controller, traditionally located in the forward path, which provides analog backup in case of a computer failure. The inputs to controllers in the forward paths would be set points. (In our simulation program we omit analog controllers for backup.)
2. The objective of the simulations was not only controller evaluation, but also familiarization with the distillation column dynamics. In the open-loop case the

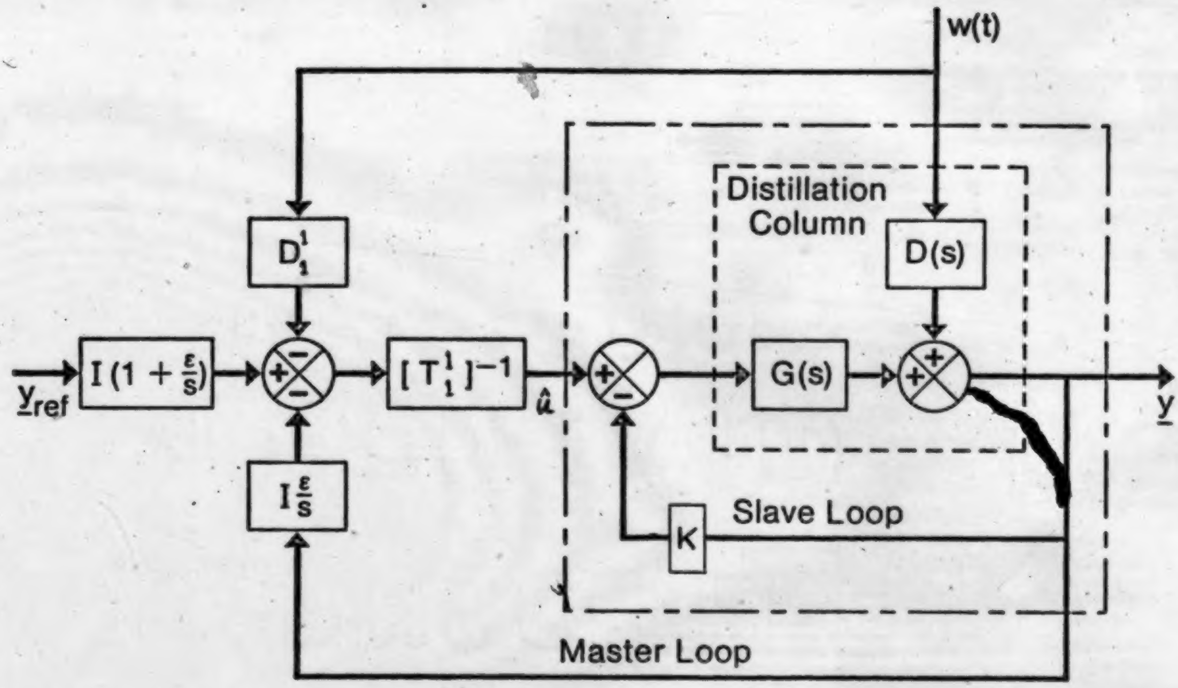

w(t)
Distillation Column
D_1^1
D(s)
$I(1+\frac{\varepsilon}{s})$
$[T_1^1]^{-1}$
$\hat{u}$
G(s)
$\underline{y}_{ref}$
$\underline{y}$
$I\frac{\varepsilon}{s}$
K
Slave Loop
Master Loop

Figure 7. Robust controller for simulation program

control inputs for the distillation column are distillate flow rate (top product) and reboiler steam flow rate. In the proportionally controlled closed loop case with conventional controllers in the forward paths, the inputs are set points for top product composition and bottom product composition. With the controllers in the feedback loops, the inputs will remain in the same engineering units (distillate flow rate and steam flow rate, as in the open loop case). We felt that this allows us to better demonstrate the linearizing effect of proportional control and also the responses of top and bottom composition to distillate flow rate and steam flow rate rather than set points gave us a better insight in the column dynamics, especially the interaction patterns.

Note that in this figure T_1' will be different from T_1' in figure 5.

In the simulation programs, the control loops are:

top loop: input = distillate flow in grams/sec.
output = top product composition in weight fraction methanol
bottom loop: input = reboiler steam flow rate in grams/sec.
output = bottom product composition in weight fraction methanol

The steady state operating conditions are:

Feed : flow rate 18.055 g/sec, 50% methanol
Distillate: flow rate 9.025 g/sec, 95.986 % methanol
Bottom : flow rate 9.030 g/sec, 4.280 % methanol
Steam : flow rate 14.577 g/sec

2.5.3 Controller Design

As was outlined in section 2.2, the robust controller was designed for linear systems. On certain kinds of nonlinear systems we have to use proportional feedback control first to linearize the system behaviour (see section 2.4). We will try to design a controller for the distillation column model and show the differences between a plant with and without a proportional slave loop.

The experiments that are required for the controller design were first performed on the open-loop plant model:

Disturbance	Top composition	Bottom composition
+5% feed = 18.958	96.221 %	8.233 %
-5% feed = 17.152	94.545 %	0.862 %
+5% dist.= 9.476	94.218 %	1.444 %
-5% dist.= 8.574	96.688 %	8.006 %
+5% steam= 15.306	96.487 %	3.779 %
-5% steam= 13.848	95.276 %	4.988 %

See also appendix B plots 1 and 2.

From these data we can design a controller

$$u = [T_1^1]^{-1} y_{ref} - [T_1^1]^{-1} D_1^1 w - [T_1^1]^{-1} \varepsilon \int_0^t (y - y_{ref})\, dt'$$

For positive disturbances,

$$[T_1']^{-1} = \begin{bmatrix} -9.80 & -9.80 \\ 89.7 & -55.9 \end{bmatrix}$$

$$\det [T_1'] = 0.00070$$

For negative disturbances,

$$[T_1']^{-1} = \begin{bmatrix} -10.2 & -10.2 \\ 86.4 & -16.3 \end{bmatrix}$$

$$\det [T_1'] = 0.00096$$

Although the determinant of the T_1'-matrix for negative changes is 37% larger than for positive changes, the controller matrices seem to be not too different. There is however a significant difference in element (2,2) of the $[T_1']^{-1}$ matrices.

The problem we indicated in section 2.3, that of sign changes in the controllers, will not arise in this control setup because the $[T_1']$ matrices are positive definite:

a matrix $\begin{bmatrix} -a_{11} & -a_{12} \\ a_{21} & -a_{22} \end{bmatrix}$ with $a_{11}, \ldots, a_{22}$ positive will always have a positive determinant, regardless of the magnitude of the entries. To change sign, one of the signs of the entries would have to change which means that the direction of one of the responses would have to change. This will not happen for normal operating conditions.

The only problem we encounter here is a large uncertainty in the magnitude of the entries, but not in the signs. In another part of this thesis we will see examples of more problematic plant behaviour.

To avoid the problems described above, we apply proportional feedback control first (see fig. 8)

The gains for top and bottom loops (k and k) were chosen somewhat arbitrarily as 95.0 and 80.0. Some runs for feed flow disturbances showed that these gains gave fast and smooth responses without oscillations (see appendix B plots 3 and 4).

The controller design experiments on the proportionally controlled plant yield:

For positive changes:

$$[T_1']^{-1} = \begin{bmatrix} -105 & -9.72 \\ 87.2 & -110 \end{bmatrix} \qquad D_1' = \begin{bmatrix} 0.00366 \\ 0.01100 \end{bmatrix}$$

For negative changes:

$$[T_1']^{-1} = \begin{bmatrix} -105 & -9.91 \\ 112 & -104 \end{bmatrix} \qquad D_1' = \begin{bmatrix} 0.00388 \\ 0.00975 \end{bmatrix}$$

As we can see, the differences between positive and negative controller matrices are fairly small, especially in the diagonal terms, so the objective of "plant linearization" has been achieved to our satisfaction.

The controller we will use is:

$$u = [T_1']^{-1} \{y_{ref} - D_1' w - \varepsilon \int_0^t (y - y_{ref})dt' \} - K_p y \qquad (35)$$

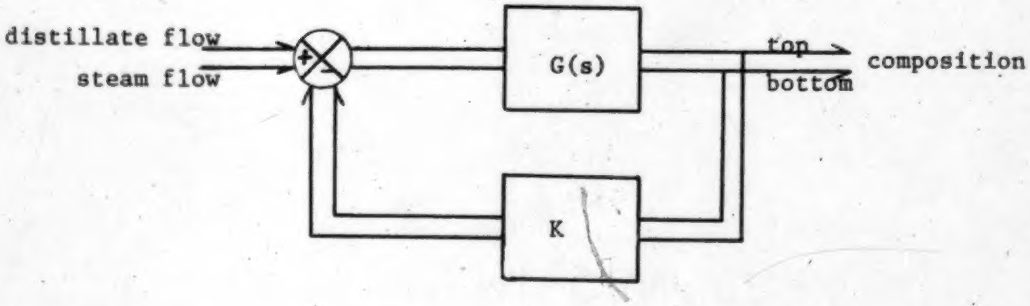


Figure 8 Plant with slave control loop

where:

$$[T_1']^{-1} = \begin{bmatrix} -105 & -9.80 \\ 99.5 & -107 \end{bmatrix} \qquad D_1' = \begin{bmatrix} 0.00377 \\ 0.01040 \end{bmatrix}$$

the average of positive and negative experimental results, and

$$K_p = \begin{bmatrix} 95 & 0 \\ 0 & 80 \end{bmatrix}$$

This controller was implemented in a discretized version with sample time 128 secs.

2.5.4 Robust Feedback-Feedforward Control

The next stage after design and implementation of the controller is the tuning, done by manipulating epsilon in equation (35) . The tuning was done for a -20% feed flow disturbance because the primary objective is regulatory

control of the distillation column.
There are well tuned conventional multiloop PI and PID controllers available that are tuned for the same disturbance.
The tuning and performance comparison was based on the calculation of the Integral Absolute Error (IAE) for 192 minutes (96 samples).

For robust control without feedback ($D_1' = \underline{0}$) the tuning results are listed in table 1.
The best results were achieved for epsilon = 0.0008

If we include feed forward action in the robust controller, the tuning results are as outlined in table 2.

We see that in this case the control is best if the robust controller has feed forward control only. Since the distillation column model is completely noise free and the parameters are time invariant, feed forward control is ideally suited to handle load disturbances. Obviously we have to compromise to get realistic results. We chose epsilon = 0.0003 to continue the simulations for the robust feedback feed forward controller.

2.5.5 Comparison to PI controllers

To evaluate the performance of the robust controller we use well tuned conventional multiloop PI controllers on the same plant model for the same operating conditions.

Table 1.

Tuning of robust feedback controller

	IAE	
epsilon	top	bottom
0.0006	23.3	62.8
0.0008	20.9	48.7
0.0010	23.6	49.8

Table 2.

Tuning of robust feedback-feedforward controller

	IAE	
epsilon	top	bottom
0.0008	15.1	32.6
0.0004	10.2	22.9
0.0003	9.7	22.2
0.0002	9.2	21.6
0.0000	7.4	17.2

The PI controllers were tuned for a -20% feed flow disturbance, the controller constants are:

top loop: $K_P = 95$, integral time $T_I = 1000$ secs.

bottom loop: $K_P = 80$, integral time $T_I = 1450$ secs.

Table 3 gives a comparison between robust feedback and PI control.

Appendix B plots 5 to 10 show some responses for robust feedback and PI control.

To evaluate the robust feedforward feedback controller we

Table 3.

Robust feedback control vs. PI control

	robust		PI	
disturbance	top	bottom	top	bottom
+20% feed flow	17.1	51.4	19.0	57.5
-20% feed flow	20.9	48.7	21.5	48.9
+1% top composition	4.4	6.8	6.2	25.5
-1% top composition	9.3	12.5	6.6	13.0
+1% bottom composition	5.4	14.3	3.7	10.6
-1% bottom composition	4.0	10.8	2.9	8.6
	61.1	144.5	59.9	164.1
	205.6		224.0	

used a PI-plus-feedforward controller for comparison. In this controller we used the same type of feedforward action as in the robust controller, i.e. we added to the multiloop PI controllers the term $-[T_1']^{-1}D_1'w$, where w is the feed flow rate and D and $[T_1']^{-1}$ are the same as in equation (35) . The performance comparison is given in table 4.
Appendix B plots 11 and 12 show responses for -20% feed flow disturbances.

In the robust feedback case we find that the robust controller is slightly better than the PI controller except for set point changes in the bottom composition. The overall control performance however is slightly better for the robust controller.

Table 4.

Robust feedback-feedforward vs. PI+feedforward

	robust		PI	
disturbance	top	bottom	top	bottom
+20% feed flow	6.5	21.7	7.2	22.5
-20% feed flow	3.8	11.7	13.1	29.7
+1% top composition	3.8	11.7	6.2	25.5
-1% top composition	6.3	11.2	6.6	13.0
+1% bottom composition	3.8	11.4	3.7	10.6
-1% bottom composition	3.6	9.8	2.9	8.6
	33.7	88.0	39.7	109.9
	121.7		149.6	

If we add feedforward action to the controllers, as shown in table 4, the gap widens. The robust feedback-feedforward controller performs better than the PI-feedforward controller for all load and set point changes except set point changes for the bottom composition, where the difference is minimal.

If we retune the PI controller after adding feedforward control, we find that the controller performance for a load change improves with increasing integral times (i.e. decreasing integral action). In fact the absence of noise and parameter drift in the plant model allows us to drop the integral action completely if we have a good feedforward controller. But then the PI controller performance for set point changes deteriorates sharply. We have not found any one PI controller with an overall performance surpassing the robust controller performance.

2.5.6 Addition of Derivative Action

It might be possible to improve the controller performance by adding derivative action to tthe controller. Now the control algorithm (35) will be modified to:

$$u = [T_1']^{-1} \{y_{ref} - D_1' w - \epsilon \int_0^t (y - y_{ref}) dt'\} - K_P y - K_D \dot{y} \qquad (36)$$

where

$$K_D = \begin{bmatrix} k_{d_1} & 0 \\ 0 & k_{d_2} \end{bmatrix}$$

and $k_{d_1} = (K_1 T_{d_1})/ISEC$, T_{d_1} is the derivative time for the top loop

We basically used the same robust controller as in section 2.5.5, but we added the derivative action and tuned the derivative times for a -20% feed flow disturbance. For this case the optimum derivative times were 110 secs. (top loop) and 180 secs. (bottom loop). For comparison we used multiloop PID-plus-feedforward controllers, tuned for a -20% feed flow disturbance:

top loop: $K_p = 95$ $T_I = 2500$ secs $T_d = 80$ secs.

bottom loop: $K_p = 80$ $T_I = 3000$ secs $T_d = 120$ secs.

See table 5 for the performance comparison.

Appendix B plots 13 to 18 show load disturbances and set point changes for PD-robust feedback feedforward control and for PID-plus-feedforward control.

Again we see that, as before, the overall performance of the

Table 5.

PD-robust + FF vs. PID+FF

disturbance	robust top	robust bottom	PID top	PID bottom
+20% feed flow	5.9	19.0	6.1	18.9
-20% feed flow	7.1	17.4	7.9	18.1
+1% top composition	4.4	11.8	5.0	45.2
-1% top composition	5.5	10.2	3.6	22.5
+1% bottom composition	2.7	10.8	2.9	9.1
-1% bottom composition	2.7	9.7	2.5	10.2
	28.3	78.9	28.0	124.0
	107.2		152.0	

robust controller is better than the conventional PID controller.

2.5.7 Conclusions

The previous sections proved that the thesis "Robust control is at least as good as or better than conventional PI(D) control" is justified, even for a nonlinear plant.

For the more important load changes, which would occur in regulatory controlled situations, the robust controller surpassed the conventional controller in all simulations.

For set point changes we saw that for top composition changes generally the robust controller was better whereas for bottom composition changes the PI(D) controller was slightly better.

The fact that the simulation programs are completely free of noise and parameter drift posed a problem in the tuning. A well designed feedforward controller will bring the output variables so close to their set points that integral action is hardly necessary to reduce the offset, or, the best integral action is no integral action.

Rather than spending much effort and time on building noise and parameter drift into the simulation programs we decided to use the results as they are because:

1. we rather spend some extra time on the more realistic pilot plant evaluation as outlined in chapter 4.
2. Both robust and PI(D) controllers suffered from the same problem so the comparison is still valid.

Not reflected in the simulation results is the tuning effort which was considerably less for the robust controller with only one tuning constant (epsilon) vs. the PI controller with 4 tuning constants.

The results as presented in this chapter justify and strongly encourage a more realistic pilot plant evaluation of the robust controller.

3. Development of a Control System

3.1 Introduction

In order to apply the robust controller as described in the previous chapter to a pilot plant distillation column we need to develop a control system, i.e. putting the hardware together and developing the software. Since small and powerfull computers have become available there has been a tendency to use these mini or micro computers in a network to increase flexibility and reliability [19,20,22,23].

This chapter describes the requirements for a process control system of the type involved in this study. Both hardware and software requirements of the computer control system are considered.

In section 3.2 the hardware configuration of the control system is described, together with how it ties in with the existing instrumentation and the distillation column. This is followed by a description of the logical setup of the complete controlled system, i.e. which tasks are situated in which computer and why. The development of the microcomputer software is described next.

The software control system has been designed to be flexible and modular. This aspect of the system design is also described in this chapter, i.e. the amount of effort involved in making changes in the programs. The chapter is concluded with an evaluation of the newly implemented total control setup.

3.2 Requirements for the Control System

It is important to consider the main requirements for a control system software package for a University research environment as opposed to an industrial system [18-20].

The main requirements are:

- perform simple conventional control (PID);
- provide sufficient data acquisition capabilities;
- provide an operator-process interface via a CRT console
- easily changeable control subroutines;
- no limitations (except program size) on the complexity of control algorithms that can be implemented, i.e. any algorithm that is Fortran programmable and does not require excessive computer memory can be used;
- easy on-line changeability of I/O-parameters in case of a system reconfiguration.

The differences with an industrial system are mainly:

1. An industrial system is generally limited to a number of fixed control algorithms.
2. Industrial systems place heavy emphasis on alarm display and acknowledgement.
3. In an industrial system the operator is not required to make on-line changes in I/O-configuration.
4. Operator interfacing is done through functional push buttons on a specially designed console rather than a regular teletype keyboard and a terminal.

The requirement of on-line changeability of I/O-parameters also assumes that all instrumented measurement signals are physically connected to the computer's I/O facilities so that reading those signals requires only software changes.

3.3 Hardware Configuration of Control System

To get an understanding of the system configuration we need to consider the instrumentation of the distillation column.

All flows are determined by measuring differential pressures across orifice plates. The differential pressure signals are converted to 3-15 psi air signals by d/p-cells (differential pressure cells). The 3-15 psi air signals are normally routed to both pneumatic analog controllers and P/I (air pressure to electric current) converters. The currents from the P/I converters are converted to voltages which are hooked up to the analog inputs of the LSI-11.

All temperatures are measured by (type J) thermocouples. The electrical signals from the thermocouples are connected to both a Honeywell temperature recorder and a low level analog input module in the LSI-11.

Pressures and liquid levels are measured with d/p-cells. The 3-15 psi air signals from the d/p-cells are connected to P/I converters. The currents from the P/I converters are, after conversion to voltages, connected to the LSI-11 analog inputs.

The top product composition is measured with a gas chromatograph (GC) which automatically samples once every three minutes. The GC-analysis is being done by an HP 1000 computer. This computer produces a GC-report which is sent to the LSI-11 (through an RS-232 line).

All controlled variables on the distillation column are flow rates. These flow rates are controlled by conventional pneumatic control valves which get their input signals from analog controllers.
The set points for the analog controllers are pneumatic signals from either manual valves in case of a local flow or level control loop, or from I/P (electric current to air pressure) converters in case of computer controlled loops. The I/P converters get their inputs (electrical currents 10-50 mA) from Current Output Stations (COS) which in turn get their inputs (0-10 Volts) from the analog outputs of the LSI-11. See figure 9 for a schematic diagram with one control loop.

The LSI-11 is a 16 bit word length microcomputer with 28K words of read/write memory [21]. This microcomputer forms the center of a fairly powerful system which also contains IO-devices (interface modules for high level analog in, low level analog in, digital in, digital out, analog out and serial line interface modules) and mass storage devices (floppy disk system).

Compared to a conventionally controlled plant, the LSI-11 microcomputer replaces analog controllers and chart recorders. It also provides new possibilities that do not exist in a conventionally instrumented plant or a loop, such

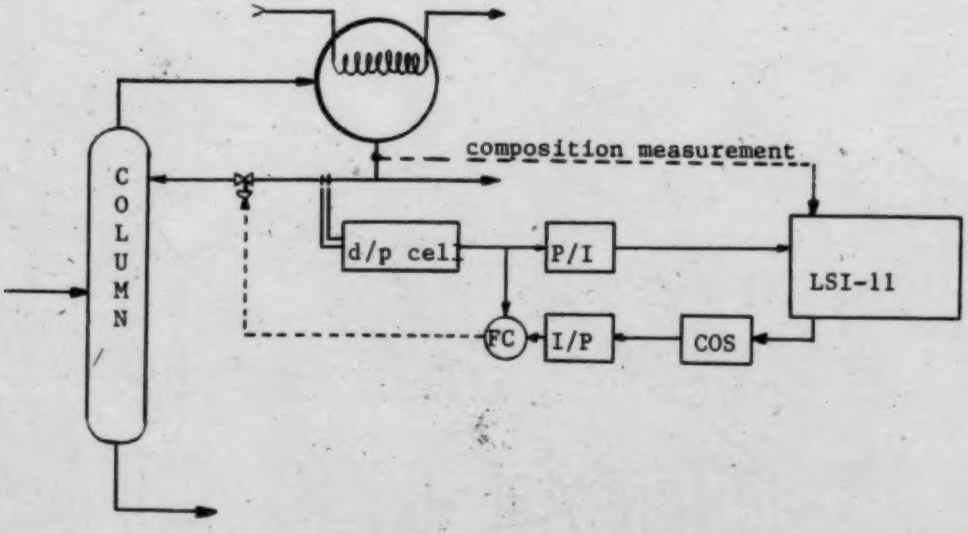


Figure 9. Distillation column with instrumentation for top loop

as composition control on the bottom loop since the microcomputer can handle an ASCII GC-report (see next section), that a pneumatic controller cannot. Another new possibility is the use of complex control algorithms which are very difficult to realize or implement using analog controllers.

The total cost of the computer system is approximately C$ 12,000 (1979). This number includes CPU, memory, a dual floppy disk drive and all process and terminal interfaces.

3.4 Overall Data Acquisition and Control System

The overall configuration of the microcomputer control system is shown in figure 10. This figure is also useful in describing the various measurement systems and links between the microcomputer and the HP-1000.

The distillation column with the LSI-11 system can be considered as a totally independent plant. The only "external" device needed is the GC computer for the bottom product composition measurement. Any failure in any of the other multi-user computer systems will not affect the distillation column.

The LSI-11 takes care of all process input, control and process output, operator interfacing and historical data storage. Except for the bottom product composition all process inputs are read directly from the distillation column instrumentation. All output signals are sent directly to the distillation column instruments.

The bottom product composition is determined from a gas chromatogram by the GC-computer. This computer produces a GC-report which is intended to be printed on a normal terminal. For our control applications the GC-report is being sent to a serial line interface module in the LSI-11. Then the report will be analyzed to find the bottom product composition.

The data acquisition is set up to send one line of data per sample interval to a floppy disk file. The line of data consists of the sample number, time, relevant process

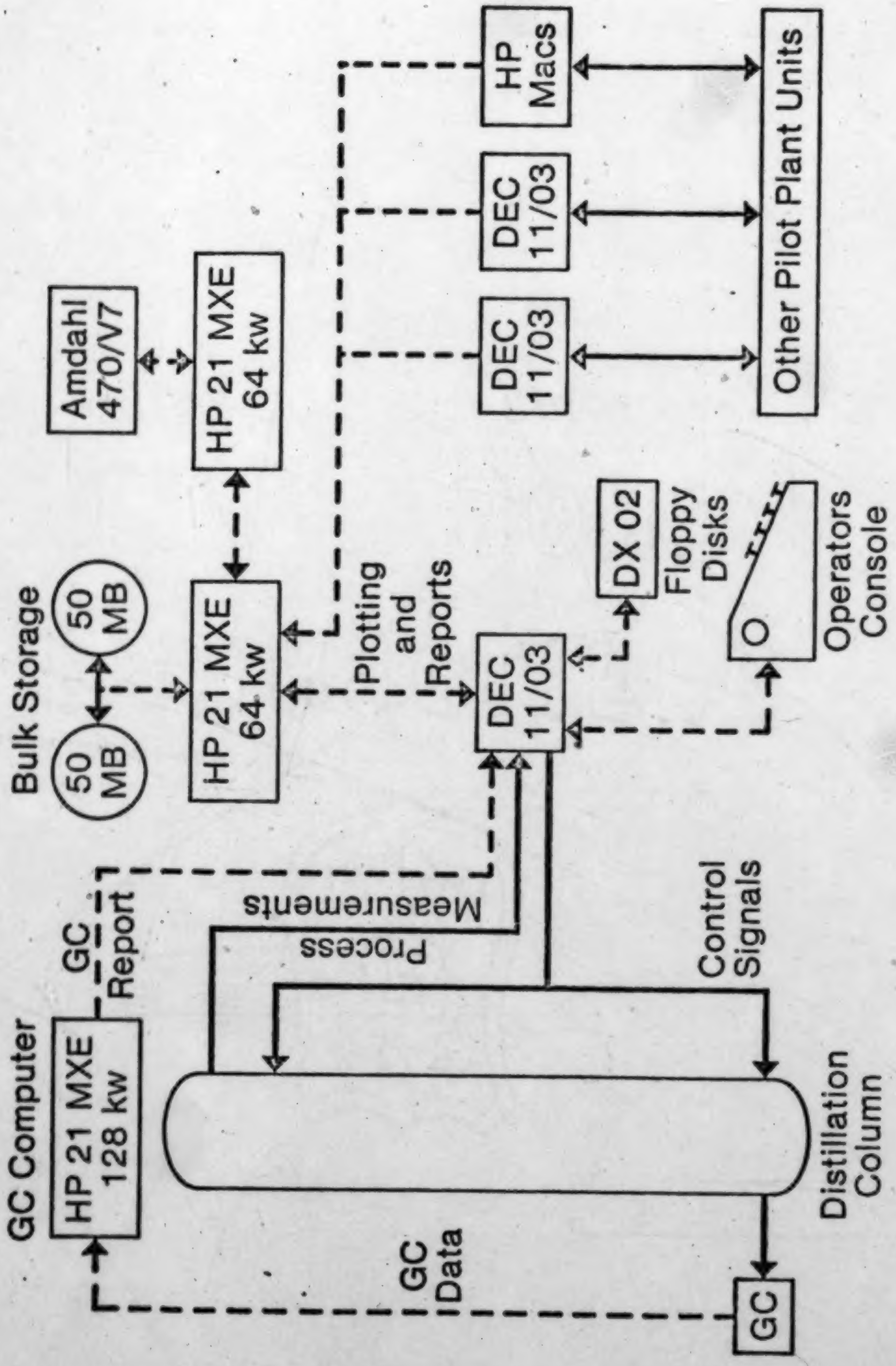

GC Computer
HP 21 MXE 128 kw
GC Report
Bulk Storage
50 MB
50 MB
HP 21 MXE 64 kw
Amdahl 470/V7
HP 21 MXE 64 kw
Plotting and Reports
DEC 11/03
DX 02
Floppy Disks
Operators Console
DEC 11/03
DEC 11/03
HP Macs
Other Pilot Plant Units
Process Measurements
Control Signals
Distillation Column
GC Data
GC

Figure 10. Schematic diagram of the distributed network of mini and micro computers and the interface with the computer controlled distillation column

readings and controller outputs. It is possible to store all data for two weeks of uninterrupted column operation on one floppy disk (approximately 900 blocks of 128 words each).

There exist data transmission facilities between the LSI-11 and the higher level HP 1000 minicomputers. This line can be used to transmit data to the HP 1000 for plotting and report generating. Future applications of supervisory and optimizing control at a higher level are also possible. There is also a data transmission link between the department's HP system and the University's Amdahl maxicomputer which allows even more sophisticated plotting and report generation.

In previous studies IBM 1800 and HP 1000 computers were used for control , in the latter case the process IO was handled by an HP-Macs system. Because of the fact that the LSI-11 is a dedicated system whereas the previous ones were multi-user systems, the reliability has significantly increased since there is no danger anymore of other users bringing the computer down.

For safety reasons there is still a "crash program" running in the HP 1000 system which takes readings of all relevant process variables, parallel to the LSI-11 (not shown in figure 10). If variables exceed safe operating limits, the crash program automatically shuts down the distillation process by tripping the main power and air switches.

In the previous paragraphs we saw that the "lower level tasks" (control and data acquisition), which require a high degree of reliability, are located in a totally dedicated microcomputer, which also takes care of presentation of data to the operator. "Higher level" off-line tasks such as appendix B plotting and report generation that do not require a high integrity system are located in a more sophisticated multi-user system that offers a wider array of user services. In this set up we tried to optimize the use of computer facilies to obtain integrity, flexibility and sophisticated user facilities, the main reasons for the current trend towards distributed computer systems [22,23].

3.5 Development of the Control Software

3.5.1 Introduction

To meet all the requirements for the control system, the microcomputer has to perform two different types of tasks:

a. tasks directly related to process control and data acquisition such as input, control calculation and send the valve signals;

b. tasks related to operator interfacing, i.e. the presentation of data to the operator and the changing of parameters by the operator.

The major differences from a software point of view are that the process control tasks are time critical, they should be performed every sample time at the right moment and in the

proper sequence.
On the other hand, the operator should be able to enter changes in the parameters whenever he likes and also list data whenever he likes.
Also, the process control and data acquisition cycle should have a high degree of integrity, maintaining a controlled situation in a plant is more important than data presentation to the operator. Operator mistakes should affect the control performance as little as possible.

To achieve the desired control integrity and operator interface independence, several of the advanced features the LSI-11 and its operating system (RT-11) are described [24-26].

The software was written in two parts, called a *foreground job* and a *background job*. (RT-11 feature). See also figure 11.

The high priority *foreground job* consists of a mainline foreground program and several subroutine libraries. Its tasks are to:

- perform the actual control, i.e. read inputs, calculate control and send outputs;
- take care of data acquisition on floppy disks;
- perform the program timing, i.e. keep track of the time and schedule the control cycle every sample time.

The lower priority *background job* consists of a mainline background program and several subroutine libraries. Its tasks are to take care of the operator input

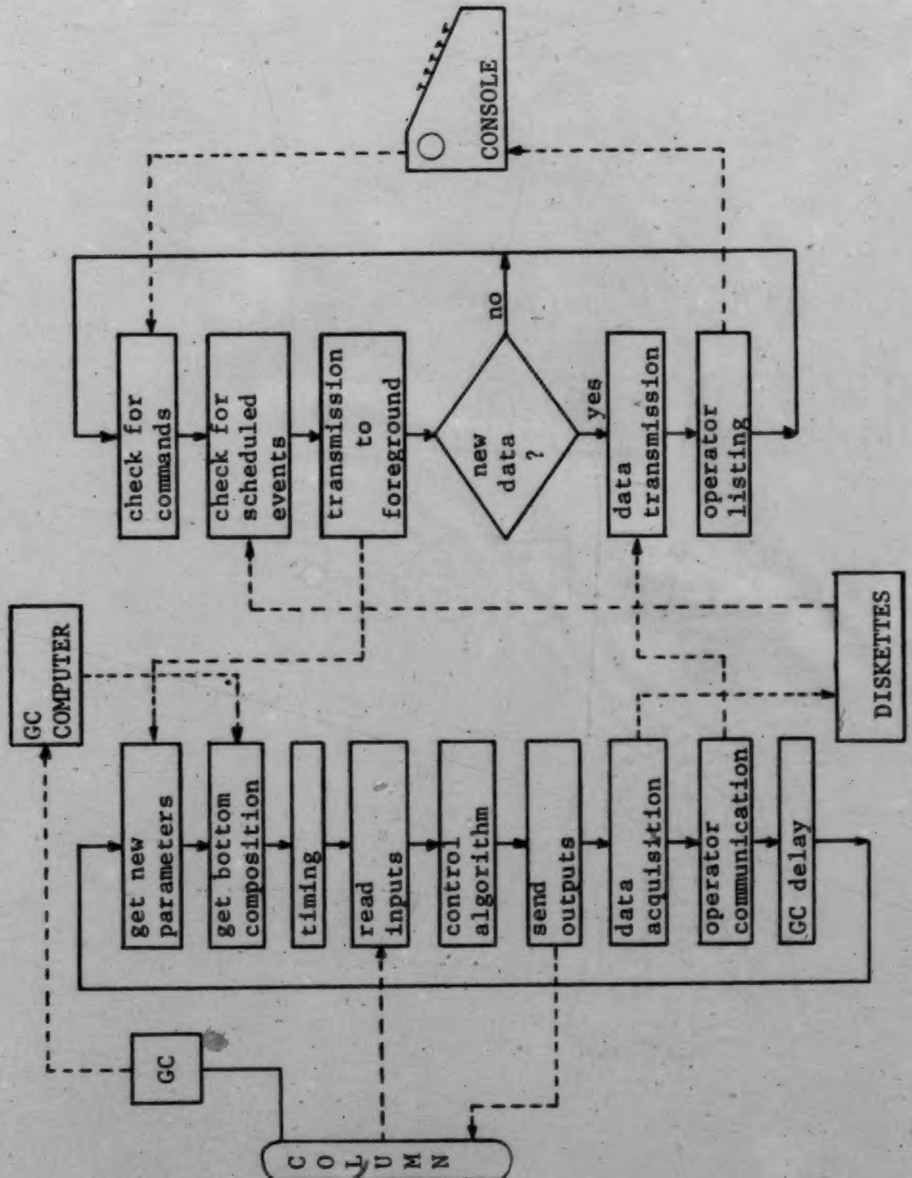

CONSOLE
check for commands
check for scheduled events
transmission to foreground
new data ?
no
yes
data transmission
operator listing
DISKETTES
GC COMPUTER
get new parameters
get bottom composition
timing
read inputs
control algorithm
send outputs
data acquisition
operator communication
GC delay
GC
COLUMN

Figure 11. Overall flow diagram for foreground and background programs

and output, that is to:

- print relevant process data every sample time and/or on demand;
- interpret and implement operator commands.

Although there is communication between the background and foreground programs, it is not necessary for the programs to run simultaneously. The setup was chosen this way to enable the operator to issue commands and interpret IO-without interfering with the process control. An operator I/O mistake that would cause the program to abort, will in this case only abort the background program without affecting the control performance.

We will first look at the way data is handled in the programs and the use of common areas. This will be followed by a detailed description of the foreground and background programs. Data acquisition on floppy disks will be discussed in the next section and we will give an overview of the operator commands. The last section will be a guideline for future users who want to tailor the system to their own needs.

3.5.2 Buffer Usage in Foreground and Background Programs

Most of the data transmission between the programs and subroutines is being handled by labelled COMMON areas. The reasons for using COMMON instead of subroutine arguments are:

1. saving of memory space;

2. all parameters and data for transmission have to be assembled in an integer buffer. To put real values in an integer buffer we use EQUIVALENCE statements. Since EQUIVALENCEd variables may not appear as subroutine arguments, one has to use COMMON.

There are four COMMON areas:

BLK1: The I/O-table.. This is the table containing all Input-Output vectors.

Dimension IOTAB(8,12)

I/O-vectors: Each of the 12 I/O-vectors in the I/O-table is 8 words long. The layout of a vector is:

MNEMONIC	VALUE	INFO	SPAN	ZERO

MNEMONIC: (words 1 and 2) 4 characters, e.g. TOPC for TOP product Composition. An unused vector starts with the characters UN. Any vector starting with UN will be ignored by input and output routines.

VALUE: (word 3) an integer number representing millivolts. For an output vector the value would be between 0-10000 (mVolt). For an input the value would be between 1000-5000 (mVolt). So the resolution for an input reading is 1/4000.

INFO: (word 4) Two bytes containing information about the type of I/O-vector. The layout is as follows:

CHANNEL NUMBER	AS SHOWN BELOW ↓

16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1

Low byte: bit 1 = 1 for input, 0 for output
2 = 1 for analog, 0 for digital
3&4 = 0 for linear input
1 for square input
2 for square root input
3 for temperature input
5-8 = not used

High byte: bits 9-16 = channel number 0-256

SPAN: (words 5 and 6) Real value giving the span of the signal. This is necessary for conversion to engineering units.

ZERO: (words 7 and 8) Real value giving the zero of the signal, necessary for conversion to engineering units.

example: the vector for the steam flowrate input:

MNEMONIC: STFL
VALUE : 1000-5000
INFO : channel #0, SQRT Analog In
bit pattern 0000000000001011
SPAN : 22.5002
ZERO : 0.0

See the documentation of subroutine ENG for an explanation of SQRT, SPAN and ZERO.

BLK2: The controller parameters table.

Dimension CPTAB(40)

This is a forty word array containing all control parameters. These are all the parameters that are required by the control algorithm.

The CP-table is EQUIVALENCEd to the following parameters:

word	*parameter*	*size*	*meaning*
CPTAB(1)	ISEC	1	sample interval in seconds
CPTAB(2)	ITYPE	1	type of control
CPTAB(3)	YREF(2)	4	set points for top and bottom composition
CPTAB(7)	KP(2)	4	P- gains for top and bottom loops
CPTAB(11)	TI(2)	4	integral times for top and bottom loops
CPTAB(15)	TD(2)	4	deriv. times for top and bottom loops
CPTAB(19)	T1I(2,2)	8	matrix for robust control
CPTAB(27)	D1(2)	4	feed forward control matrix
CPTAB(31)	EPS(2)	4	tuning constants for robust control
CPTAB(35)	SIAE(2)	4	integral absolute errors for both loops
CPTAB(39)	not used		
CPTAB(40)	not used		

BLK3: The program parameters table

Dimension PPTAB(10)

This is a ten word array containing program parameters for the foreground program:

word	*parameter*	*size*	*meaning*
PPTAB(1)	ITIME(2)	2	clock time in internal clock

format

PPTAB(3)	NSAMP	1	sample number
PPTAB(4)	ISNORE(2)	2	waiting time for GC in minutes and seconds
PPTAB(6)	not used		
....	....		
PPTAB(10)	not used		

BLK4: Data transmission flags

FLAGD = 1 byte flag for data transmission

FLAGP = 1 byte flag for parameter transmission

3.5.3 The Foreground Program

After program startup, the program first enters the initialization phase where:

- initial values for all parameters will be read from the initialization file (INIT.DAT);
- the operator enters the name of the file to be used for data acquisition.

Then the program jumps into a continuous loop:

1. If any new parameter values are sent by the background program, those new values are inserted in place of the old values, e.g. change from P to PI control.
2. The GC report which is being sent by an HP 1000 computer is read and analyzed. The result of the analysis is a number representing the bottom composition.
3. A timing routine waits for the next sample time before continuation, then,
4. the GC is triggered to get a new sample, and

5. all specified analog input points are read. This reading gives us numbers for all relevant process variables.
6. The control algorithm calculates new settings for the control variables, typically reflux flowrate and steam flowrate.
7. The specified Current Output Stations are addressed and switched to the proper settings to set the control valves.
8. The relevant data is written to floppy disk file
9. All data and parameters are transmitted to the background program
10. While the GC is analyzing the next sample, the program will wait (approx. 2.5 minutes).
11. Return to 1.

Figure 12 illustrates the flow diagram of this foreground program.

3.5.4 The Background Program

After program startup, we first enter the initialization phase:

- initial values for all parameters will be read from the initialization file (INIT.DAT);
- the "event file" (EVENT.DAT) will be opened and the time for the first scheduled event will be read (see section 3.5.6.6 on event scheduling.

Then the program branches into a continuous loop and does the following tasks:

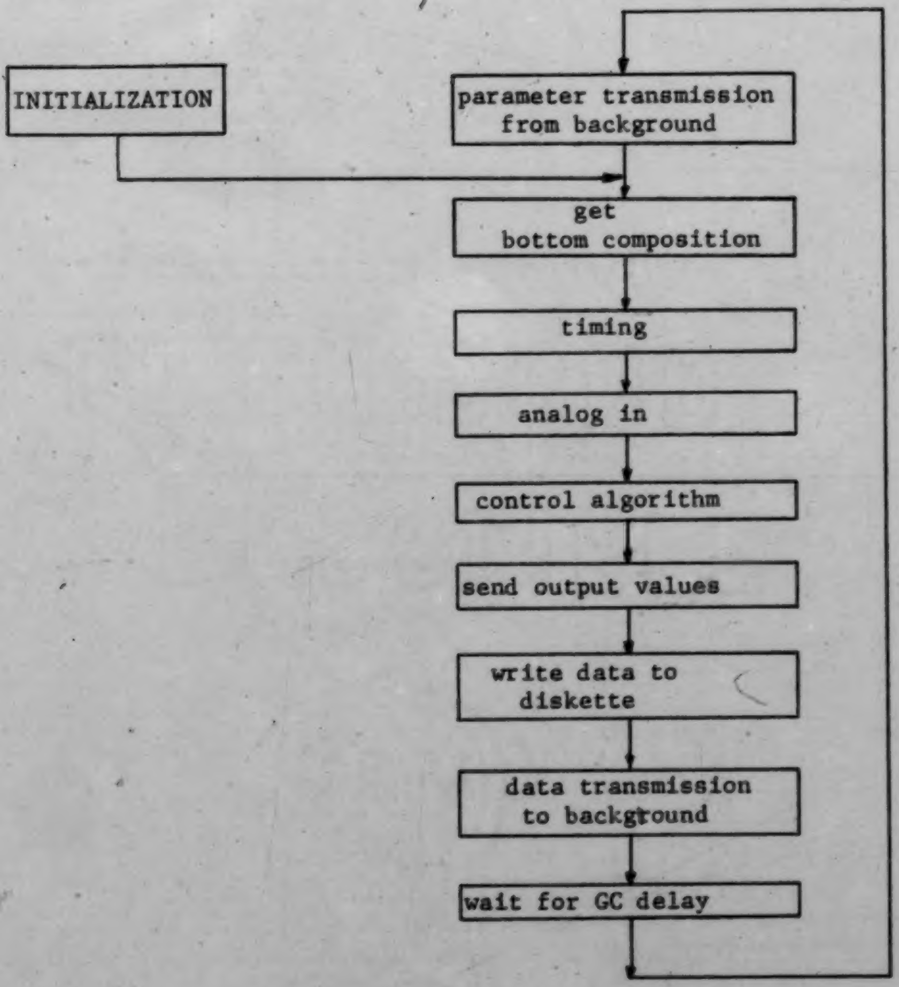

INITIALIZATION
parameter transmission from background
get bottom composition
timing
analog in
control algorithm
send output values
write data to diskette
data transmission to background
wait for GC delay

Figure 12. Flow diagram for foreground program

1. Check for commands and scheduled events and services them.
2. If there is no new data from the foreground program, goes back to 1.
3. If there is new data from the foreground program, it puts the data in the tables.
4. Prints relevant data on the operators console.
5. Returns to 1.

The form of the algorithm is illustrated in figure 13.

3.5.5 Data Acquisition on Floppy Disks

Every sample time, after the input-control-output algorithms, the foreground program writes relevant data to floppy disk. First, however, a disk file must be defined, and this can be done in two ways:

1. *Immediately after starting the foreground program*, the program will print an asterisk (*) in the left margin on the console. The operator is expected to enter a filename:
   - First enter a **CTRL-F** to direct data to the foreground,
   - then enter the filename, e.g. **DY1:DA3132.DAT** (DAta file, opened on 3 - 13, 2nd file of the day) The operator can of course enter any name he likes.

   Entering **TT:** in place of the filename will direct the data to the console.
2. *While the program is running.* By adding 20 to the value of ITYPE (see section 3.5.6.4), the operator can tell

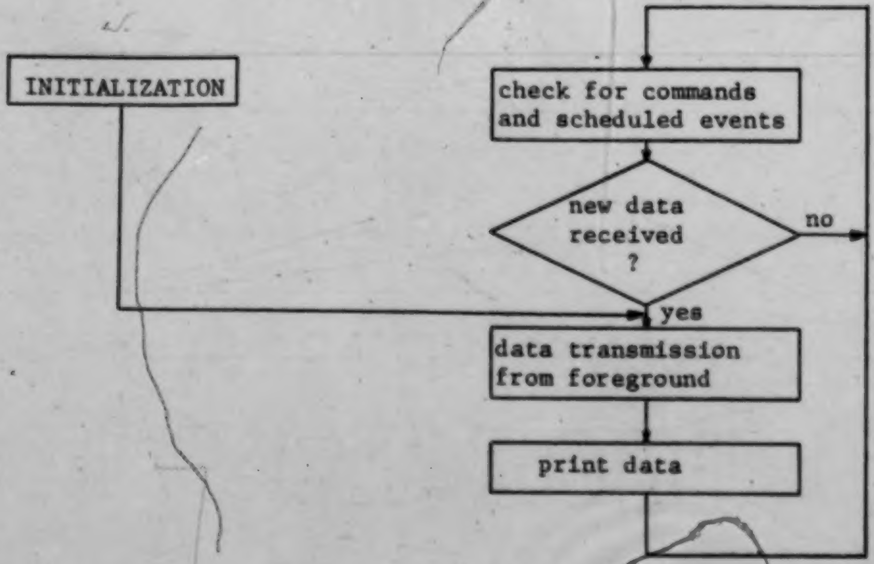

INITIALIZATION
check for commands and scheduled events
new data received ?
no
yes
data transmission from foreground
print data

Figure 13. Flow diagram for background program

the foreground program to close the old data file and open a new one. ITYPE will be automatically reset to its old value.

Example: say ITYPE=3 and the operator wants to close the old data file and open a new one. By doing so he saves the data in the old file. The operator sets ITYPE to 3+20 = 23. Just before next sample moment, the following will appear on the console:

```
FILE CLOSED, ENTER NEW FILENAME
*
```

The operator replies CTRL-F "filename"

Since the foreground program hangs up while waiting for the filename, it is advisable to enter the new filename immediately to prevent losing a sample.

Every sample time one record of data is being written to the disk file, this record presently consists of 11 integer numbers:

word
1. = sample number
2. = time, minutes past midnight
3. = time, seconds past minutes past midnight
4. = top product composition, a number 1000 - 5000 (mVolts)
5. = distillate flowrate,............. 1000 - 5000
6. = reflux flowrate,................. 1000 - 5000
7. = feed flowrate,................... 1000 - 5000
8. = steam flowrate,.................. 1000 - 5000
9. = bottom product composition,...... 1000 - 5000
10. = reflux flowrate setting,......... 0 - 10000
11. = steam flowrate setting,.......... 0 - 10000

To save different data on file one would have to change the 'write' command in the foreground program.

3.5.6 Operator Commands

Assuming the system is set up properly and the proper floppy disks are inserted in the disk drives, then one can proceed and run the programs as follows:

3.5.6.1 Run the foreground program:

```
COMMAND: FRUN DY1:FOR2/N:2000
```

This command assumes that the foreground program file FOR2.REL is situated on the floppy disk in drive 1. The additional space that has to be allocated (in our case 1500 words) can be calculated from the formula on page 4-6 of the Advanced Programmers Manual [24].

Immediately after this command, the program responds with an asterisk (*) in the left margin. The operator is now expected to enter a filename for data acquisition. First enter **CTRL-F** (while pressing the control key, press F) to direct the filename to the foreground program.

Example (operator entries bold)

```
*
CTRL-F.......go to foreground program
DY1:DATA.DAT CR (Carriage Return) filename
CTRL-B.......return to background
```

Now the foreground program should be running. The operator, however, will not see any listing on the console until he runs the background program.

3.5.6.2 Run the background program

```
COMMAND: RUN DY1:BACK3
```

This command assumes that the background program file BACK3.SAV is situated on the floppy in drive 1.

Note: One of the things the background program does is to take care of the event scheduling. The program will look for a file EVENT.DAT in drive 0 and read the first line in this file, i.e. the time for the first scheduled event. If the operator does not want event scheduling, the first line in the event file has to be 00:00, but the event file must exist. Look in the next section for details on event scheduling.

Now, every sample time a concise listing will appear on the console. This consists of two lines of numbers indicating sample number, time, process variables and set points. Before every fifth listing a line of labels will be printed on the console:

```
 #    TIME       TOPC     DIST     REFX    FFLO    STFL     BOTC
30  10:50:31   95.376   10.318    7.021   18.250  14.125    5.008
                                  7.203           14.100


31  10:53:31   95.258   10.125    7.210   18.105  14.086    4.996
                                  7.312           13.996
```

3.5.6.3 The LIST command

There are five printing options in the background program:

1. *Concise*

   This consists of two lines of numbers giving an overview of the most important process variables and set points.

2. *I/O-table*

   This option lists the contents of the I/O-table, i.e. 12 I/O-vectors. One vector would look like:

   TOPC 3000 LIN AI 11 12.8572 89.1673

See the chapter on buffer usage (3.5.2) for a detailed description of I/O-vectors.

3. *CP-table*

   This option lists the contents of the CP-table.

4. *PP-table*

   This option lists the contents of the PP-table

5. *All tables*

   This option lists the I/O-table, CP-table and PP-table.

The operator can get a listing on the console by entering

L return

and the program will reply with CONCISE(1), IO TABLE(2), CP TABLE(3), PP TABLE(4) OR ALL(5)

and the operator can enter his choice, e.g.

3 return

This will produce a listing of the CP-table.

Instead of L **return**

the operator could have entered **L3 return**

which would have produced an immediate listing of the CP-table.

3.5.6.4 The CHANGE command

Instead of just listing the table, the operator can also enter changes in the tables. To do this the operator has to enter the numbers of the parameters he wants to change and the program will then ask for the new value of the indicated parameter.

For example the operator wants to change the set point for the top product composition from 95% to 96%. The dialog is then:

C return

the program replies with

```
CONCISE(1), IO-TABLE(2), CP-TABLE(3), PP-TABLE(4) OR ALL(5)
```

The set points are controller parameters, and are to be found in the CP-table. The operator enters

3 return

then the program will list the CP-table and ask

```
ENTER PARAMETER NUMBER(S) TO CHANGE:
```

In the CP-table listing the numbers are listed beside the parameters. The set points are parameter 3, so the operator enters

3 return

and the program will reply

```
NEW SETPOINTS YREF:
```

Now the operator has to enter both set points (top-bottom):

96.0,5.0 return

The CP-table will be listed with the new set points YREF and the question

```
SATISFIED? (Y/N)
```

Entering **Y return** will cause the new parameters to be sent to the foreground program for implementation.

Entering **N return** will cause a reply by the program

```
CONCISE(1), IO-TABLE(2), CP-TABLE(3), PP-TABLE(4) OR ALL(5)
```

then the operator can enter the table number and make additional changes or undo the first change.

If the operator wants to enter changes in the I/O-table, he will have to enter two parameter numbers, indicating the position of the parameters to be changed. The listing of the I/O-table looks like:

```
   MNEM.  VALUE  INFO  CH.#   SPAN    ZERO


 1  TOPC  3000  LIN  AI  11  12.857  89.167
 2  DIST  3000  SQRT AI   8  15.205   0.0
 3  REFX  1461  SQRT AI   1  22.975   0.0
 4  FFLO  3000  SQRT AI   2  25.777   0.0
 5  STFL  2400  SQRT AI   0  22.500   0.0
 6  BOTC  1625  LIN  DI   0  32.000   0.0
 7  U1SP  1152  SQRT AO   2  22.975   0.0
 8  U2SP  3494  SQRT AO   3  22.500   0.0
 9  FEED  5013  SQRT AO   1  25.777   0.0
10  UNU2     0  LIN  DO   0 100.0     0.0
11  UNU3     0  LIN  DO   0 100.0     0.0
12  UNU4     0  LIN  DO   0 100.0     0.0
```

If the operator wants to enter a -25 % change in feed flowrate, he will enter for parameter numbers

2,9 return

where 2 indicates that the operator wants to change the VALUE and 9 is the number of the I/O-vector (see the left hand margin in the table listing). The program will reply

ENTER NEW VALUE, 0-10000:

and the operator would enter

2845 return (i.e. output = 2845 mVolts) which will set the feed flow to approximately 13.5 grams per second (see explanation in next section).

3.5.6.5 Feed flow rate changes

The operator can set the set point for the feed flow controller by manipulating the VALUE in the I/O-vector labelled FEED.

Say the controller has a bias of -0.25 grams/sec, i.e. if we send a signal corresponding to 18 grams/sec to the controller, the actual flow will be regulated to 17.75 grams/sec. This so called "bias" is in reality the sum of biases of the instruments between the LSI-11 and the pneumatic controller (analog output module in LSI-11, Current Output Station, I/P-converter and pneumatic analog controller).

Given this -0.25 grams/sec bias, we want to calculate the output voltage for the feed flow so that the actual feed will go to 18 grams/sec.

For a flow rate the calculation is:

$$\left(\frac{\text{desired value - bias}}{\text{SPAN}}\right)^{2} * 10000 \ \text{mVolts}$$

Thus, to get a feed flowrate of 18 grams/sec, the output has to be

$$\left(\frac{18.0 + 0.25}{\text{SPAN}}\right)^2 * 10000 = 5013 \text{ mVolts}$$

3.5.6.6 Event scheduling

To enable the operator to schedule events ahead of time, which facilitates column operation without the presence of an operator, an event scheduler has been incorporated in the background program. This basically consists of the subroutine SCHED which is being called by subroutine SERVE, called by the mainline background program. The event scheduling requires the operator to enter parameter changes in a file instead of on the system console. The filename is **EVENT.DAT** in floppy drive 0. Theoretically, an operator can schedule weeks or even months ahead, depending on the size of the event file.

Every scheduled event consists of three lines of code in the event file:

1. The first line contains the time at which the event is to take place. We use a 24 hour clock with the format **hrs:min**, e.g. 1 am is **01:00** and 4.15 pm is **16:15**. This format should always be used in the event file.
2. The second line contains the address of the parameter that is to be changed. The format for this line is: **table number, parameter number 1, parameter number 2**

e.g. the feed flowrate setting is parameter 2,9 in the I/O-table, so the address of the feed flowrate setting is

2,2,9   table number 2, 2nd entry in the I/O-vector, I/O-vector number 9.

The set points are the third parameter in the CP-table, so their address is

3,3,0   table number 3, 3rd entry

3. The third line contains the new value of the parameter that is to be changed, e.g.

2845 for a feed flowrate

95.0,4.0 for the set points at 95% (top) and 4% (bottom)

Example

Say it is 4 pm and the distillation column is operating at steady state (feed flowrate 18 grams/sec, top composition 95%, bottom composition 5%). The operator wants to schedule the following events:

-at 6 pm the feed flowrate should drop to 13.5 grams/sec

-at 1 am (next day) the set point for the bottom composition should drop to 4% (wt% methanol).

EVENT.DAT

```
18:00
2,2,9,
2845
01:00
```

```
3,3,0,
95.0,4.0,
00:00
```

NOTE: After the last scheduled event there should always be a line 00:00 to signal the scheduler that it should stop reading from the file. Without this line an input error will result.

3.5.7 Modifying the Programs

If a user wants to make modifications in the programs to accomodate his own type of control, he will have to write his own control subroutine (CONTR). All other changes in the subroutines are made necessary by the fact that a user may want to use different controller parameters, ergo the only changes that have to be made are in the subroutines that use the controller parameters: to list them (LISCP), to change them (CHANCP) and to read their initial values from a file (mainline foreground and background programs).

We will look at what exactly has to be done:

a. A user will probably have different parameters in his control algorithm. These parameters will have to be declared in the subroutine and be accomodated in the CP-table, i.e. the parameters have to be EQUIVALENCEd to words in the CP-table.

   In the user written subroutine CONTR the only requirements are:

   1) the COMMON blocks BLK1 (I/O-table) and BLK2

(CP-table) have to be inserted;

2) if the input values for the flow controllers are calculated in engineering units, the values have to be converted to machine units (mVolts) with a CALL INGE (see listing of CONTR and INGE in appendix A)

The user can put his parameters in the CP-table with an EQUIVALENCE statement, similar to the way it was done in the standard control algorithm (see listing of CONTR in appendix A)

b. If the user wants to have the same flexibility and convenience in listing and changing his own controller parameters, some changes will have to be made in subroutines LISCP and CHANCP in the subroutine library SUBBA.FOR.

1) LISCP

The only requirement is that COMMON /BLK2/CPTAB be declared. The rest of the subroutine consists of an EQUIVALENCE statement and a WRITE statement. The user has to EQUIVALENCE his own parameters to the CP-table and write them in any format he likes.

2) CHANCP

In CHANCP we want to read new values for controller parameters. The user again has to declare COMMON /BLK2/CPTAB and EQUIVALENCE it to his own parameters, which have to be numbered and refered to by the numbers chosen by the user.

The simplest way of modifying CHANCP is to replace the EQUIVALENCE and READ statements in the standard subroutine (see listing of CHANCP in appendix A)

c. If the initialization sections of the foreground and background programs read all initial values for the parameters from a file INIT.DAT, the user has to make the following changes in the initialization sections of both foreground and background programs:
   1) Replace the EQUIVALENCE statement for the CP-table
   2) Change the READ statement which reads data from file INIT.DAT to read the user's parameters.
   3) Of course the user also has to replace the present parameter values in file INIT.DAT by the values for his own parameters.

d. If the user wants to include other parameters in the data acquisition, he will have to include those parameters in the WRITE statement in the foreground program which writes data to disk file. (See listing of FOR2.FOR in appendix A)

The amount of effort required for modifying the control subroutine depends on the complexity of the user's algorithm, which replaces the current robust algorithm. All other changes should require not more than four hours of coding.

If the user needs more controller parameters than will fit into the present CP-table (40 words), the user will have to increase the size of the CP-table. The complication that arises here is the parameter transmission between foreground and background programs. The user will have to increase the size of the transmission buffers as well. See the program listings (appendix A) and [24].

3.6 Evaluation of the Control System

The computer system we used to implement the control algorithm on proved to be easy to use and reliable. The fact that we have a dedicated single user system makes it in some respects more flexible and dependable, especially where the communication with the GC computer is concerned since the data transmission is synchronous (the GC does not wait for an LSI signal).

After two weeks of testing and modifying, all experimental runs on the distillation column were done during 7 weeks of continuous operation. During those 7 weeks we experienced 9 system failures, 3 caused by untraceable system errors, 4 caused by GC failures and 2 caused by erroneous operator entries. The system failures just caused the control values to "freeze", which didn't lead to instability or a crash of the distillation process.

4. Experimental Robust Control of the Distillation Column

4.1 Introduction

Using the control system that was developed in the previous part of this thesis we can now evaluate the performance of Davison's robust controller on a real-life plant: the department's pilot plant distillation column.

We will have a brief look at distillation column control in general and describe the set up for *two quality energy balance control* which we used for the implementation of robust control.

Then we will describe the distillation column that was used to implement the control on. An extensive description of the control hardware can be found in part 2 of this thesis.

Before we implement a controller, we have to perform the steady state experiments as outlined in the theory (section 2.2). These experiments will show that a cascaded control arrangement with proportional control in the slave loop is necessary.

Then the design and implementation of the robust controller is straightforward and we can evaluate its performance and compare it to conventional multiloop PID controllers.

One of the problems encountered in controlling the distillation column was the long time delay in the bottom loop. The conventional strategy for dealing with time delays is the use of derivative action in controllers. We tried to improve the robust controller performance by including derivative action in the slave loop of the cascaded robust controller. We will see that robust control in a cascaded arrangement indeed does offer some advantages over conventional PI(D) control.

4.2 Distillation Column Control

Separation of light and heavy components in a mixture is one of the most important operations in chemical industry, especially in hydrocarbon fuel processing. As a direct result continuous distillation units are considered to be the work horses of most chemical plants and certainly of oil refineries [27].

The technique of distillation is well developed and major gains through better distillation column design and construction are not expected. Nevertheless the rapidly increasing energy costs have provided a strong incentive to try to increase the efficiency of distillation columns. This has led to an increased interest in control [28-30] and the introduction of more and more advanced control strategies, all aiming at reducing the operating costs.

Especially illustrative is optimal control where the control action is directly aimed at minimizing a "cost function". Most control stategies however, have as primary objective operating the column closer to the specification limits with as justification an expected reduction in operating costs.

Traditionally, several basic control strategies have been used, depending on product, objectives, etc. [27]:

1. Mass balance control. Here we keep two of the three flow rates (or ratios) constant. This method can be used when the feed composition is practically constant and there is one product which is much more important than the other.
2. Temperature-and-pressure control. The principle is to keep temperature and pressure on some tray in the column at constant values. For binary mixtures at bubble point the composition will then be constant too. For mixtures consisting of homologous series of hydrocarbons this method generally works too, provided one does not try to fractionate closely boiling components.
3. Single quality control. We measure a local composition directly, independent of the pressure. The difference with 2) lies mainly in the different interations between the various loops.

All the above methods try to keep a distillation column close to an ideal operating point. If we would want to change the specifications for, say, the bottom product, but not for the top product, the change in operating point would not be straight forward. The situation is more clear if we use two-quality-control, with composition analyses of the top and bottom products. With conventional multiloop PID controllers, we have two direct control loops. In the case of the departments pilot plant column (see section 4.3) these loops are:

top loop - in which the top product composition is measured
    (in the condenser), and
    the reflux flow rate is manipulated,

bottom loop - the bottom product composition is measured
    (sample from the reboiler), and
    the reboiler steam flow rate is manipulated.

One quality controller (bottom) in fact adjusts the vapour flow in the column, the other (top) the liquid flow.

If we look at the manipulated variables in the column, this control scheme can be called an *energy balance control scheme* because we manipulate the energy flows to the column (steam and reflux). The opposite would be a material balance control scheme where we manipulate product flows to regulate compositions.

Tests performed by Lieuson [32] showed that on this column for the operating conditions we used, an energy balance scheme gives better results.

Past experiences with the department's distillation column [30-40] have shown that controlling this column constitutes a realistic test for any control algorithm, where nonlinearities, time delays and interactions pose the problems that are to be overcome.

4.3 Distillation Column Equipment

The 22.8 cm diameter, 8 plate, pilot scale distillation column used in this study has been used in a number of previous control studies [30-40]. Each plate at a spacing of 30.5 cm is fitted with 4 bubble caps. Operating conditions are adjusted to provide a separation of the 50 weight percent methanol-water mixture into top and bottom compositions of about 95 and 5 weight percent methanol respectively. A detailed summary of typical steady state operating conditions are given in table 6.

A schematic diagram of the column, which is completely instrumented with conventional industrial control valves, controllers and transmitters is shown in figure 14. As can be appreciated, only the principal control instrumentation pertinent to the objective of this control study has been shown. Since the objective is to maintain top and bottom compositions at their specified values, both streams are equipped with instrumentation for composition analysis. Top composition, although continuously monitored using a temperature compensated in-line capacitance probe, is sampled at the same rate as the bottom composition signal. Bottom composition is measured using a HP 5720A gas chromatograph (GC). Liquid sampling of the bottom product stream is under the control of one of three HP 1000's of the Department's distributed computer network of mini and micro-computers. Analysis of the chromatogram, on a 180 seconds cycle, is also performed by this node of the

Table 6.
Typical steady state operating conditions

Feed flow rate	18.0 g/s
Reflux flow rate	8.8 g/s
Steam flow rate	13.6 g/s
Top product flow rate	9.0 g/s
Bottom product flow rate	9.0 g/s
Feed composition	50.0 % (weight % methanol)
Top composition	95.0 % (weight % methanol)
Bottom composition	5.0 % (weight % methanol)

distributed network wich then transmits a signal to a 'local' node of the network, a DEC 11/03 microcomputer. The control sample interval for both loops is equal to 3 minutes, as a unit integer multiple of the 3 minute measurement delay time that occurs in the GC analysis. The schematic configuration of the pilot plant units and the distributed computer network is shown in figure 9 (page 48). As can be seen from the diagram, implementation of the control algorithm and data acquisition, except for the GC composition measurement , is performed using the DEC 11/03. Operator communication to the DEC 11/03 is by means of a CRT terminal. See section 3.4 for a more detailed description of the system.

4.4 Experiments for Controller Design

As was outlined in the description of the theory, the robust controller design technique was intended for use on a linear system. In section 2.3 we have shown that a nonlinear

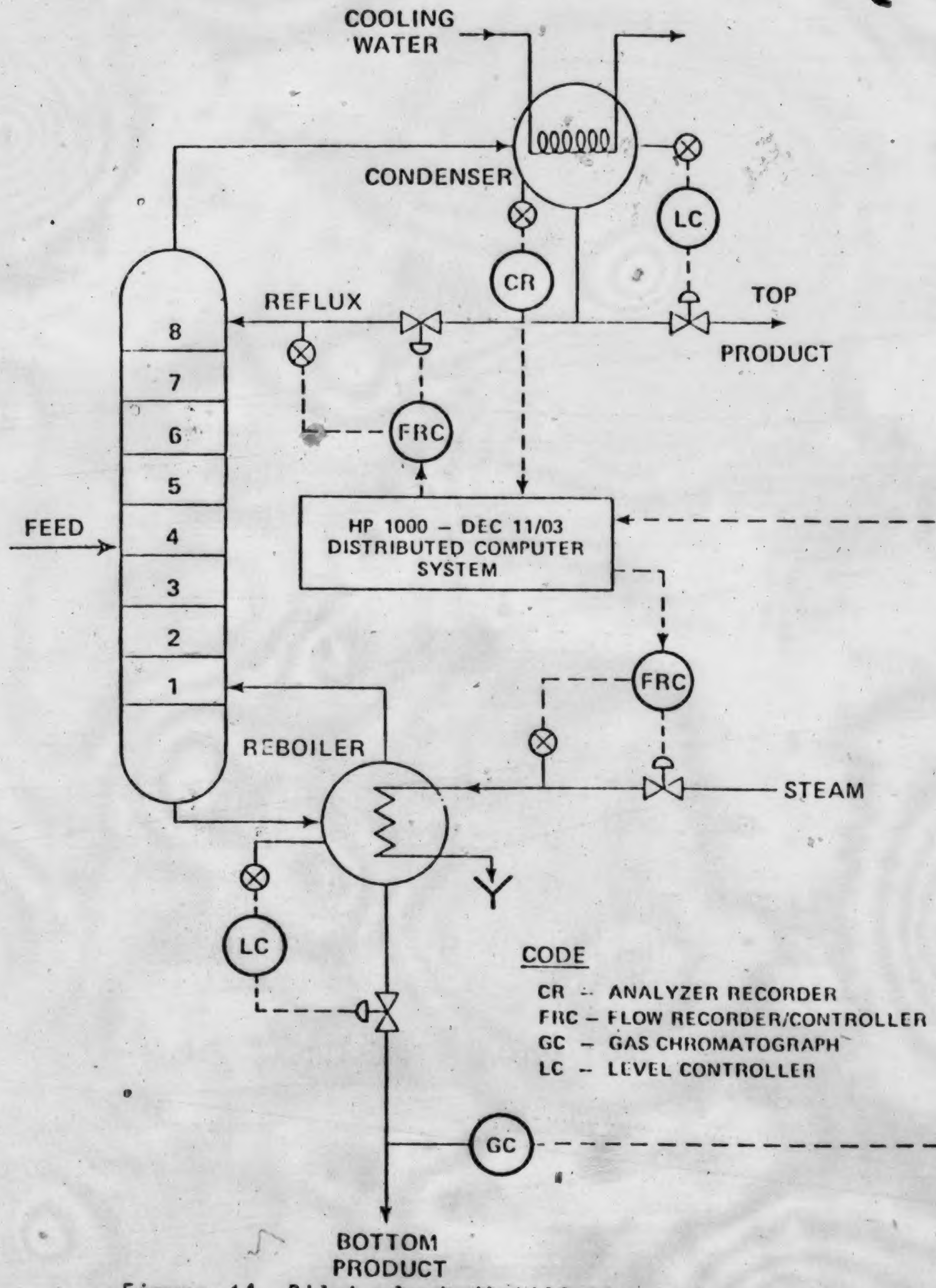

COOLING WATER
CONDENSER
LC
CR
REFLUX
TOP PRODUCT
8
7
6
5
4
3
2
1
FRC
FEED
HP 1000 – DEC 11/03 DISTRIBUTED COMPUTER SYSTEM
FRC
REBOILER
STEAM
LC
CODE
CR – ANALYZER RECORDER
FRC – FLOW RECORDER/CONTROLLER
GC – GAS CHROMATOGRAPH
LC – LEVEL CONTROLLER
GC
BOTTOM PRODUCT

Figure 14. Pilot plant distillation column with instrumentation

plant may yield different controllers for the same operating points, depending on whether we use positive or negative inputs. This is also the case for the distillation column under consideration.

For positive step inputs (+10% reflux flow rate and +8% steam flow rate) we find:

$$[T_1'] = \begin{bmatrix} 0.826 & -1.400 \\ 3.000 & -4.400 \end{bmatrix} \qquad [T_1']^{-1} = \begin{bmatrix} -7.785 & 2.477 \\ -5.308 & 1.462 \end{bmatrix} \qquad (37)$$

For negative step inputs (-10% reflux flow rate and -8% steam flow rate) we find:

$$[T_1'] = \begin{bmatrix} 1.077 & -0.539 \\ 2.366 & -6.154 \end{bmatrix} \qquad [T_1']^{-1} = \begin{bmatrix} 1.149 & -0.101 \\ 0.442 & -0.201 \end{bmatrix} \qquad (38)$$

Appendix B plots 19 and 20 show responses for positive and negative changes in reflux flow rate.

We see that (37) and (38) have opposite signs, so we cannot construct a controller for this operating point. The differences in (37) and (38) are such that averaging does not make sense since we cannot even determine the signs in the controller matrix from above results. Therefore we first apply multiloop proportional feedback control to the distillation column, as was suggested in section 2.4 .

The primary purpose of the P-control is to "linearize" the plant, i.e. to give as much as possible the same responses for positive and negative set point changes. We want to avoid the behaviour as outlined in 1.3 where we showed that the robust design technique on a nonlinear plant can give us two totally different controllers for the same set of operating conditions.

The choice of K_p is not very important and only a few test runs are necessary to obtain acceptable values. Since values for PID-controllers were available, we started with the gains from the PID-controllers (3.571 top and -0.576 bottom). Plot 21 (appendix B) shows the responses to +20% change in feed flow rate. The response is fast with a little bit of overshoot in top and bottom composition. A full series of experiments as outlined in [1] (see section 2.2.2) for both positive and negative inputs and disturbances shows that the objective of "plant linearization" is fairly well satisfied:

positive changes:

$$[T_1']^{-1} = \begin{bmatrix} 1.506 & -0.032 \\ -1.649 & 1.359 \end{bmatrix} \qquad D_1' = \begin{bmatrix} -0.0549 \\ 0.7985 \end{bmatrix} \tag{39}$$

negative changes:

$$[T_1']^{-1} = \begin{bmatrix} 1.306 & -0.069 \\ -0.979 & 1.488 \end{bmatrix} \qquad D_1' = \begin{bmatrix} -0.0216 \\ 0.5696 \end{bmatrix} \tag{40}$$

average:

$$[T_1']^{-1} = \begin{bmatrix} 1.406 & -0.051 \\ -1.314 & 1.424 \end{bmatrix} \qquad D_1' = \begin{bmatrix} -0.038 \\ 0.684 \end{bmatrix} \qquad (41)$$

Because the response shows some overshoot which might lead to oscillatory behaviour in a robust controlled situation, we repeat the experiments with lower gains. We use 70% of the previous gains (2.5 top loop, -0.4 bottom loop). Because the offset will be larger we decrease the step size of the inputs. Plots 22 and 23 (appendix B) show the responses to +10% and -10% changes in the feed flow rates. This time we have fast and smooth responses. Calculation of $[T_1']$ -matrices after conducting all experiments shows that the linearizing effect of P-control is still satisfactory:

positive changes:

$$[T_1']^{-1} = \begin{bmatrix} 1.370 & -0.077 \\ -1.841 & 1.423 \end{bmatrix} \qquad D_1' = \begin{bmatrix} -0.026 \\ 0.935 \end{bmatrix} \qquad (42)$$

negative changes:

$$[T_1']^{-1} = \begin{bmatrix} 1.581 & 0.111 \\ -2.096 & 1.988 \end{bmatrix} \qquad D_1' = \begin{bmatrix} -0.042 \\ 0.744 \end{bmatrix} \qquad (43)$$

average:

$$[T_1']^{-1} = \begin{bmatrix} 1.476 & 0.017 \\ -1.969 & 1.706 \end{bmatrix} \qquad D_1' = \begin{bmatrix} -0.034 \\ 0.840 \end{bmatrix} \qquad (44)$$

To round off the experiments we repeat the same procedure for gain values of 50% of the first case (1.785 top, -0.288 bottom). Plot 24 (appendix B) shows the responses to +10% change in feed flow rate. The transients are smooth and slightly slower than in the previous case. There is still a reasonable amount of linearization:

positive changes:

$$[T_1']^{-1} = \begin{bmatrix} 2.770 & -0.469 \\ -5.994 & 2.983 \end{bmatrix} \qquad D_1' = \begin{bmatrix} -0.054 \\ 0.831 \end{bmatrix} \qquad (45)$$

negative changes:

$$[T_1']^{-1} = \begin{bmatrix} 1.974 & -0.116 \\ -3.212 & 1.922 \end{bmatrix} \qquad D_1' = \begin{bmatrix} -0.018 \\ 1.057 \end{bmatrix} \qquad (46)$$

average:

$$[T_1']^{-1} = \begin{bmatrix} 2.372 & -0.293 \\ -4.603 & 2.453 \end{bmatrix} \qquad D_1' = \begin{bmatrix} -0.036 \\ 0.944 \end{bmatrix} \qquad (47)$$

We found that the second case with gains 2.5 for top loop and -0.4 for bottom loop gave an acceptable amount of plant linearization combined with smooth and fast responses that will probably provide the best basis for the next step, the implementation and tuning of a robust controller.

4.5 Robust Feedback Feedforward Control

We are most interested in regulatory control, i.e. the ability of the controller to keep the output values (top and bottom product compositions) as close as possible to the desired values (set points) in the face of disturbances and parameter variations.

Traditionally, controllers for the distillation column have been tuned for feed flow disturbances while the set points were being kept constant. The robust controllers that will be described in this and the following sections will be tuned for a -25% feed flow rate disturbance. The PID controllers that were available for comparison were also tuned for this disturbance.

The controller we used was (see section 4.4):

$$u = K_p [T_1']^{-1} \{\dot{y}_{ref} - D_1' w - \varepsilon \int_0^t (y - y_{ref}) dt' \} - K_p y \quad (48)$$

where:

$$[T_1']^{-1} = \begin{bmatrix} -105 & -9.80 \\ 99.5 & -107 \end{bmatrix} \qquad D_1' = \begin{bmatrix} 0.00377 \\ 0.01040 \end{bmatrix}$$

The controller was tuned for a -25% feed flow disturbance, both with and without feed forward action (i.e. $D_1' = \underline{0}$). The tuning was done by manipulating epsilon and calculating the Integral Absolute Errors (IAE) for top and bottom product composition from the responses. See table 7 for the results. (Calculation over 1.5 hrs. = 30 samples was carried out, which was sufficient for the transient to die out)

If we look at the sum of IAEs for top and bottom product compositions, we find that the best controller setting for robust control without feedforward is epsilon = 0.15. For robust feedback feedforward control we find epsilon = 0.10 the best setting.

A full series of runs, including feed flow disturbances and set point changes for the best controller settings gave results as outlined in table 8.
Appendix B plots 25 to 30 show responses for feed flow and set point changes under robust control.
For robust control without feedforward we find the IAE values outlined in table 9.

4.6 Comparison of Robust and PID controllers

In this section we will look at the robust controller's performance relative to conventional multiloop PID controllers.
A PID controller uses a control algorithm:

$$u = K_p * E + K_I * \Sigma E + K_D * (y_o - y) \tag{49}$$

Table 7.

Tuning of robust feedback feedforward controller

	IAE without feedforward		IAE with feedforward	
epsilon	top	bottom	top	bottom
0.05	6.8	58.8	5.7	4.9
0.10	4.4	35.4	3.0	6.2
0.15	3.5	30.7	2.4	9.0
0.20	3.9	34.4	2.2	10.5
0.25	4.9	40.1	2.2	11.0

Table 8.

Load disturbances and set point changes for robust feedback feedforward control

	IAE		IAE return to steady state	
CHANGE	top	bottom	top	bottom
+25% feed flow rate	4.5	9.3	4.3	4.2
-25% feed flow rate	3.0	6.2	3.4	5.1
+1% top composition	3.1	5.7	3.4	7.2
-1% top composition	2.2	4.3	1.9	4.3
+1% bottom composition	0.8	10.2	0.9	7.4
-1% bottom composition	0.8	7.3	1.0	8.0

Table 9.

Load disturbances and set point changes for robust feedback control

CHANGE	IAE top	IAE bottom	IAE return to steady state top	IAE return to steady state bottom
+25% feed flow rate	4.3	29.8	3.7	28.3
-25% feed flow rate	4.0	28.2	3.8	24.8
+1% top composition	2.1	5.8	2.7	6.1
-1% top composition	2.9	6.0	2.0	4.7
+1% bottom composition	1.3	8.9	0.9	7.4
-1% bottom composition	0.8	7.3	1.0	7.8

where u = control value

K_p = proportional gains

K_I = integral gains = $(K_p*ISEC)/T_I$

where ISEC = sample interval in seconds

T_I = integral time in seconds

K_D = derivative gains = $(K_p*T_d)/ISEC$

where T_d = derivative time in seconds

E = error = $y_{ref} - y$

y_o = previous value of the output y

The controller we used for comparison was tuned for a -25% feed flow disturbance. The controller settings are:

Top loop : y = top product composition

u = reflux flow rate

$K_p = 3.571$, $T_I = 695.0$ secs., $T_d = 41.6$ secs.

Bottom loop : y = bottom product composition

u = reboiler steam flow rate

$K_p = -0.576$, $T_I = 771.0$ secs., $T_d = 39.5$ secs.

The performance comparison for various disturbances and set point changes is outlined in table 10 . The PI controller is the same as PID but with the derivative action set to zero. Appendix B plots 31 and 32 show -25% feed flow rate responses for PI and PID control.

The above results show that PI and PID control is slightly better than robust control. The robust controller however was designed with feedforward action in mind. It is obvious that the performance of the robust feedback feedforward controller for load disturbances will be better than a normal PID controller, due to the feedforward action. To make the comparison more realistic, we added feedforward action to the PID controller so that the PID + feedforward control algorithm is:

$$u = K_p * E + K_I * \Sigma E + K_D * (y_* - y) + K_p * [T_1']^{-1} * D_1' * FF \qquad (50)$$

FF= Feed flow rate

See equation (41) for $[T_1']^{-1}$ and D_1'.

Note that the $[T_1']^{-1}$ and D_1' matrices were designed for the values of K_p used in the PID controller (see section 4.4)

Table 10.

Load disturbances and set point changes for robust feedback control vs. PI and PID control

	IAE					
	Robust feedback		PI		PID	
CHANGE	top	bottom	top	bottom	top	bottom
+25% feed flow	4.3	29.8	2.6	25.9	2.8	27.3
-25% feed flow	4.0	28.2	2.3	23.6	2.1	21.7
+1% top comp.	2.1	5.8	3.0	11.6	3.1	10.0
-1% top comp.	2.9	6.0	2.4	6.9	3.0	8.4
+1% bottom comp.	1.3	8.9	0.8	6.1	0.7	4.4
-1% bottom comp.	0.8	7.3	0.8	5.2	1.1	5.6
	15.4	86.0	11.9	79.3	12.8	77.4
	101.4		91.2		90.2	

The performance comparison is given in table 11. Appendix B plots 33 to 35 show responses to feed flow and set point changes for PID-plus-feedforward control.

We see that the performance of the robust controller is roughly the same as the PI and PID performance. For set point changes we find that the robust controller is better for changes in top composition, worse for bottom composition changes. This can be explained from the fact that there is a strong interaction between the steam flow rate and the top product composition. The multivariable controller accounts for this interaction and thereby minimizes the undesirable change in bottom composition. In case of a set point change in bottom composition, the robust controller basically behaves as a conventional multi-loop controller since there is hardly a ripple in the top composition. Here the robust

Table 11.

Load disturbances and set point changes for robust feedback feedforward control vs. PI and PID control

	IAE					
	Robust feedback feedforward		PI +feedforward		PID +feedforward	
CHANGE	top	bottom	top	bottom	top	bottom
+25% feed flow	4.5	9.3	1.8	9.5	1.5	9.1
-25% feed flow	3.0	6.2	4.3	23.7	1.7	12.6
+1% top comp.	3.1	5.7	3.0	11.6	3.1	10.0
-1% top comp.	2.2	4.3	2.4	6.9	3.0	8.4
+1% bottom comp.	0.8	10.2	0.8	6.1	0.7	4.4
-1% bottom comp.	0.8	7.3	0.8	5.2	1.1	5.6
	14.4	43.0	10.5	62.4	11.1	50.1
	57.4		72.9		61.2	

controller loses its multivariable advantage over the PID controller.

If we look at the sum of IAEs, we see that PI and PID controllers do a slightly better job controlling the top loop, but the robust controller gives better control for the bottom loop. This means that the robust controller's advantage in incorporating interaction in the control algorithm outweighs the tuning advantages and derivative action in the PID controllers. The bottom loop is considered to be more difficult to control than the top loop because of the long time delays. The robust controller has a slight advantage over PID controllers since the IAE presents a measure of the amount of off-spec product produced by the distillation column. This is less than for PID control.

4.7 Robust Control plus Derivative Action

The major control problem in our system is posed by the long time delay in the bottom loop (7 minutes). Because of its phase lead properties derivative action has been proposed as medicine for time delays (phase lag) in conventional control schemes. It may be expected that adding derivative action to the slave loop of the robust controller will give us some improvement in the control. Since derivative action operates on the system outputs only, it is easy to add to the control algorithm. Also, derivative action does not affect the steady state behaviour of a controller, therefore we do not have to redesign the robust controller if we want to use PD control in the slave loop instead of P control.

Since we want to keep as much as possible the tuning advantage of the robust controller, the derivative action was implemented as "add-on" feature and also tuned as such, i.e. the best robust controller tuning from the previous data was taken and then only the D action was varied in the tuning runs. Again all tuning runs were -25% feed flow disturbances. The tuning results are listed in table 12 As we see the effects of derivative action are minimal. The differences in IAEs cannot be called significant. Our best choice of derivative time 40 seconds for both loops is

Table 12.

D-tuning for -25% feed flow disturbance
PD-robust feedback feedforward control

Derivative time		IAE	
top loop	bottom loop	top	bottom
10	10	4.4	6.3
20	20	4.3	7.0
30	30	4.2	5.3
40	40	4.2	5.0
50	50	4.2	6.1
10	20	3.8	7.4
20	40	4.2	7.5
40	80	4.2	8.7
80	160	3.7	8.4
160	320	4.7	10.2

somewhat based on intuition and supported by the fact that this is the same amount of derivative action as in the well tuned PID controller.

We also did some derivative tuning for PD-robust control without feedforward action but the differences between the runs were very insignificant. The comparison of the overall controller performances is given in tables 13 and 14. Appendix B plots 36 to 41 show the column responses to feed flow disturbances and set point changes for PD-robust control.

As we can see, the overall results for PD-Robust control are slightly better than P-Robust or PID control. The control improvement lies largely in the bottom loop (compare the total IAE figures for the bottom loops). This was to be expected since the derivative action can cope

Table 13.

Load disturbances and set point changes for P- and PD-robust feedback control vs. PID control

IAE

CHANGE	P-Robust top	P-Robust bottom	PD-Robust top	PD-Robust bottom	PID top	PID bottom
+25% feed flow	4.3	29.8	4.2	28.0	2.8	27.3
-25% feed flow	4.0	28.2	4.6	33.4	2.1	21.7
+1% top comp.	2.1	5.8	2.1	4.7	3.1	10.0
-1% top comp.	2.9	6.0	3.7	3.9	3.0	8.4
+1% bottom comp.	1.3	8.9	1.3	8.4	0.7	4.4
-1% bottom comp.	0.8	7.3	1.4	7.2	1.1	5.6
	15.4	86.0	17.3	85.6	12.8	77.4
	101.4		102.9		90.2	

Table 14.

Load disturbances and set point changes for P- and PD-Robust+feedforward control vs. PID+Feedforward control

IAE

CHANGE	P-Robust +feedforward top	P-Robust +feedforward bottom	PD-Robust +feedforward top	PD-Robust +feedforward bottom	PID +feedforward top	PID +feedforward bottom
+25% feed flow	4.5	9.3	4.0	2.7	1.5	9.1
-25% feed flow	3.0	6.2	4.2	5.0	1.7	12.6
+1% top comp.	3.1	5.7	2.5	3.6	3.1	10.0
-1% top comp.	2.2	4.3	3.8	4.6	3.0	8.4
+1% bottom comp.	0.8	10.2	1.7	7.0	0.7	4.4
-1% bottom comp.	0.8	7.3	1.3	7.8	1.1	5.6
	14.4	43.0	17.5	30.7	11.1	50.1
	57.4		48.2		61.2	

5

better with the long time delay we find in the bottom loop.

5. Conclusions

5.1 Controller Evaluation

The evaluation of the robust controller performance can be based on both the simulations from chapter 2 of this thesis and the experiments in chapter 4. Throughout this thesis we have been using Integral Absolute Errors (IAE) for comparison with PID controllers. Before we start drawing conclusions for the "quality of the control", some comments should be made about the interpretation of IAEs.

Firstly, the objective of control is to keep the output values as close as possible to the desired values under all circumstances. For a constant throughput, the IAE represents a measure of the amount of off-spec product produced. Since we are dealing with compositions, we can mix liquid that exceeds the specification limits with liquid that is below the specification limit to obtain an end product that is exactly on-spec. Similarly, if the composition of one of the distillation column products would follow a sinusoidal oscillation around the desired value, we would (on the average) produce no off-spec product, but the IAE would keep on increasing. Therefore it is sometimes argued that in the case of compositions as outputs, an Integral Error criteria would be better than an Integral Absolute Error criteria. Counter arguments are that oscillations in the outputs do represent problems for downstream units. This should be reflected in the performance criteria. Also the basic

objective of control is minimizing the operating costs. Oscillations in product quality and subsequent blending may produce on-spec product, but the operation represents an entropy increase and is therefore wasteful of energy [29]. The IAE does "penalize" oscillations.
Our decision to use IAE was based on these considerations and on the fact that IAE results for PID controllers were available for comparison.

Secondly, in the case of simulations the IAE values are determined by system dynamics and controller performance only. In the experimental runs we also have to contend with noise, parameter drift and unmeasurable disturbances. Therefore, small differences in IAE values between PID and Robust controller for simulation runs are the direct result of controller performance, the system dynamics being the same. But in the experimental runs, minor variations in IAE are not meaningfull for the controllers because they might be caused by external disturbances, noise, etc.

The first conclusion that should be drawn is that Davison's robust controller as such cannot deal with a system exhibiting the nonlinearities as mentioned in this thesis. Only after modifications (the addition of a slave control loop) we can expect reasonable controller performance.

Looking at the simulation results, we see that robust feedback control performs slightly better than PI control. If we add feedforward action to the controllers, the gap widens, and especially for the more important load disturbances we see a significant improvement. Adding up all IAE values for feed flow and set point changes, we find that the total IAE for robust feedback feedforward control is 19% less than for PI control. Looking at feed flow disturbances only, we find an improvement of 40%.

If we add derivative control action to both Robust and PI controllers, we find an improvement in control. Now the total IAE improvement is 29% but for feed disturbances only an improvement of 3%, ergo the improvement lies largely in set point changes.

For the experimental runs we find that, compared to PID-plus-feedforward control, P-robust feedback feedforward control gives a 6% overall improvement. Including derivative action gives control 21% better than PID-plus-feedforward. Looking at feed flow disturbances only, we find P-robust control 8% better and PD-robust 36% better than PID control.

The difference in improvements between simulations and experimental runs is caused by noise and unmeasurable disturbances in the experimental case which make integral control action necessary. In the simulations, integral action in the controller actually caused a deterioration of control performance (see section 2.5.4). As has been shown in the previous chapters, robust multivariable control can offer significant advantages over conventional control. This conclusion is also supported by [41], although this paper does not indicate any measures to counter nonlinearity problems. If we compare all controller performances that were outlined in the previous sections, we find that the best distillation column control is achieved with PD-Robust Feedback Feedforward control. Although the improvements over conventional PID control do not seem to be very significant, one should keep in mind that distillation column control is mostly regulatory control. If we look at the controller performances for feed flow disturbances only, we see that robust controllers in our case do offer some significant improvements due to the much better control of the bottom loop.

Also one should consider that the performance ot the PID controller is upgraded by the addition of feedforward control. The feedforward controller was designed with Davison's technique, so part of the credit for the good showing of the PID controller should go to the robust controller design technique.

Although it does not show up in plots and tables, we should also mention that it took only a minimal tuning effort to achieve a robust controller which equals or surpasses the performance of the conventional PID scheme. Proper tuning of a PID controller is a very time consuming and complicated task which requires a high degree of expertise from the control engineer, and the tuning process would probably yield much more off-spec product.

5.2 Recommendations for future work

A possible improvement can be made in the control software. Currently the control programs take up almost all available memory in the LSI so that the system would not accomodate a much larger control algorithm. Some subroutines can be made more space efficient without affecting the system performance. Especially the data transmission between foreground and background programs which presently require large buffers can be made more efficient if necessary.

One of the major problems in the control is the long measurement time delay in the bottom loop (3 minutes) because of the GC analysis. Improvement of controller performance could be achieved by reducing this time delay. This can be done by reducing the GC analysis from 2.5 minutes to 1.5 to 2 minutes. Next, one could choose a higher baud rate for data transmission from the GC computer to LSI which can reduce the report time from approximately 30 seconds to less than 1 second. Previously this was impossible because of limitations of the time-sharing control computer (HP 1000) which demanded a slow data transmission.

An even bigger improvent could be achieved by using the temperature on tray 1 as an indicator for bottom composition. Since we have a binary mixture at bubble point, Gibb's phase rule says that we have two degrees of freedom. Since this is an atmospheric distillation column, the pressure is fixed and by measuring the temperature we can determine the composition. The GC analysis could still be used in a cascaded arrangement to improve accuracy. By thus enabling a greatly increased sample rate at no measurement delay, a major control improvement may be expected.

References

[1] E.J. Davison, 'Multivariable Tuning Regulators: The Feedforward and Robust Control of a General Servomechanism Problem'. IEEE Trans. Auto. Control, AC-21, Feb. 1976.

[2] H. Seraji, 'Design of Proportional-Plus-Integral Controllers for Multivariable Systems'. Int. J. Control, *29*, No. 1, pp. 49-63, Jan. 1979.

[3] O.A. Sebakhy and W.M. Wonham, 'A Design Procedure for Multivariable Regulators'. Automatica, *12*, pp. 467-478, 1976.

[4] K.J. Astrom and B. Wittenmark, 'On Self-Tuning Regulators'. Automatica *9*, 185-199 (1973).

[5] K.J. Astrom, U. Borisson, L. Ljung and B. Wittenmark, 'Theory and Applications of Self-Tuning Regulators'. Automatica, Vol. 13, 457-476, 1977.

[6] B. Wittenmark, 'Stochastic Adaptive Control Methods', a survey. Int. J. Control *21*, 705-730, 1975.

[7] E.J. Davison, 'The Robust Control of a Servomechanism Problem for Linear Time-Invariant Multivariable Systems'. IEEE Trans. Aut. Contr., Vol. AC-21, No. 1, Feb. 1976.

[8] E.J. Davison and A. Goldenberg, 'Robust Control of a General Servomechanism Problem: The Servo Compensator'. Automatica, Vol. 11, pp. 461-471, 1975.

[9] D.P. Turtle and P.H. Phillipson, 'Simultaneous Identification and Control'. Automatica, *7* , pp. 445-453,

1971.

[10] K.J. Astrom and P. Eykhoff, 'System Identification - A Survey'. Automatica, 7, pp. 123-162, 1971.

[11] B.A. Francis and W.M. Wonham, 'The Internal Model Principle of Control Theory'. Automatica, Vol. 12, 457-465, 1976.

[12] H.G. Kwatny and K.C. Kalnitsky, 'On Alternate Methodologies for the Design of Robust Linear Multivariable Regulators'. IEEE Trans. Auto. Control, AC-23, 930-933, 1978.

[13] L. Ljung, 'Analysis of Recursive Stochastic Algorithms'. IEEE Trans. Auto. Control, AC-22, 1977.

[14] J.C. Doyle, 'Robustness of Multiloop Linear Feedback Systems', Honeywell Systems and Research Center, 1978

[15] D.G. Fisher and J.F. Kuon, 'Comparison and Evaluation of Multivariable Frequency Domain Design Techniques', Proc. 4th IFAC symp. on Multivariable Technological Systems, Fredericton, Canada, July 1977.

[16] A.G.J. MacFarlane and J.J. Belletrutti, 'Characteristic Locus Design Method', Automatica, no. 9, 575 - 558, 1973.

[17] W.Y. Svrcek, 'Dynamic Response of a Binary Distillation Column', Ph. D. Thesis, University of Alberta, Dept. of Chem. Eng., 1967.

[18] R.J. Bibbero, 'Microprocessors in Instruments and Control'. John Wiley & Sons, 1977.

[19] T.J. Harrison, 'Minicomputers in Industrial Control'.

Instrument Society of America, 1978.

[20] J.P. Shunta and W.F. Klein, 'Microcomputer Digital Control What it Ought to Do'. ISA Trans., *18*, No. 1, 1979.

[21] 'Microcomputer Processors', Digital Equipment Corporation, Maynard, Massachusetts, 1978.

[22] D.G. Fisher, S.L. Shah, 'A Distributed Network for Process Control and University Research'.

[23] R.E. Dessy, 'Computer Networking, a Rational Approach to Lab Automation', Analytical Chemistry, *49*, no. 13, Nov. 1977.

[24] 'RT-11 Advanced Programmers Guide', Digital Equipment Corporation, Maynard Massachusetts, 1977.

[25] 'RT-11 System Users Guide', Digital Equipment Corporation, Maynard, Massachusetts, 1977.

[26] 'RT-11 Fortran IV Users Guide', Digital Equipment Corporation, Maynard, Massachusetts, 1977.

[27] O. Rademaker, J.E. Rijnsdorp, A. Maarleveld, 'Dynamics and Control of continuous Distillation Units', Elsevier Scientific Publishing Company, New York, 1975.

[28] C.O. Schwanke, T.F. Edgar and J.O. Hougen, 'Development of Multivariable Control Strategies for Distillation Column'. ISA Trans.,*16*, No. 4, 1978.

[29] F.G. Shinskey, 'Distillation Control for Productivity and Energy Conservation', McGraw-Hill, 1977.

[30] C.B.G. Meyer, R.K. Wood and D.E. Seborg, 'Experimental Evaluation of Analytical and Smith Predictors for

Distillation Column Control'. AIChE Journal, *25*, 24-32 (1979).

[31] R. Bilec and R.K. Wood, 'Multivariable Frequency Domain Controller Design for a Binary Distillation Column'. Reprint for 86th AIChE meeting in Houston, Texas, April 3, 1979.

[32] H. Lieuson, 'Experimental Evaluation of Self Tuning Control of a Binary Distillation Column', M.Sc. Thesis, University of Alberta, Dept. of Chem. Eng., 1980.

[33] V.A. Sastry, D.E. Seborg and R.K. Wood, 'Self-Tuning Regulator Applied to a Binary Distillation Column'. Automatica, *13*, 417-424 (1977).

[34] B.M. Chanh, 'Control of a Binary Distillation Column: Effect of Sensor Location', M.Sc. Thesis, University of Alberta, Dept. of Chem. Eng., 1971.

[35] R.G. McGinnis, 'Multivariable Control of a Binary Distillation Column', M.Sc. Thesis, University of Alberta, Dept. of Chem. Eng., 1972.

[36] M.W. Berry, 'Terminal Composition Control of a Binary Distillation Column', M.Sc. Thesis, University of Alberta, Dept. of Chem. Eng., 1973.

[37] W.C. Pacey, 'Control of a Binary Distillation Column. An Experimental Evaluation of Feedforward and Combined Feedforward-Feedback Control Schemes', M.Sc. Thesis, University of Alberta, Dept. of Chem. Eng., 1973.

[38] P.W. Liesch, 'Decoupled Feedforward-Feedback Control of a Binary Distillation Column', M.Sc. Thesis, University

of Alberta, Dept. of Chem. Eng., 1974.

[39] C.B.G. Meyer, 'Experimental Evaluation of Predictor Control Schemes for Distillation Column Control', M.Sc. Thesis, University of Alberta, Dept. of Chem. Eng., 1977.

[40] R. Bilec, 'Modeling and Dual Control of a Binary Distillation Column', M.Sc. Thesis, University of Alberta, Dept. of Chem. Eng., 1980.

[41] E.J. Davison, P. Taylor, J. Wright, 'On the application of Tuning Regulators to Obtain Robust Feedforward-Feedback Multivariable Controllers for an Industrial Heat Exchanger'. For submission to IEEE, June 1979.

6. APPENDIX A: Description of programs and subroutines

NAME: FOR2 LOCATION: FOR2.FOR

DATE: 791114 AUTHOR: G.W.M. COPPUS

ABSTRACT: Mainline foreground program, see description below

CALLING SEQUENCE:

CALLED BY:

CALLS: ASSIGN........System Subroutine
CLOSE.........System Subroutine
IQSET.........System Subroutine
GTIM..........System Subroutine
MTATCH........System Subroutine
MTGET.........System Subroutine
MTSET.........System Subroutine
IRCVDF........System Subroutine
CVTTIM........System Subroutine
ISDATF........System Subroutine
ISLEEP........System Subroutine
GLUE..........Assembler Subroutine
DOUT..........Assembler Subroutine
GETBC.........Fortran Subroutine
TIMER.........Fortran Subroutine
DATAC.........Fortran Subroutine
CONTR.........Fortran Subroutine
PUT...........Fortran Subroutine
COMP..........Fortran Subroutine
COMD..........Fortran Subroutine

DESCRIPTION:

Inialization section

After the declaration statements, file INIT.DAT on drive DY0: will be opened. It contains all initial values for variables in the three tables (I/O-table, CP-table, PP-table).

The values are read and the file is closed again. A separate subroutine (GLUE) puts the real values of SPAN and ZERO into the integer arrays of the I/O-table.

IQSET sets aside extra space for queue elements needed for timer and IO calls.

The GC link (see description of subroutine GETBC) is set up as a terminal link. CALL MTATCH attaches a new terminal to the system, reserves buffer space for IO, etc. MTGET returns the terminal characteristics and MTSET sets the new characteristics (e.g. inhibit echos, see GETBC).

The GC remote start switch is closed by default at system power up. The program sets the switch to its normal (open) position.

The program opens a file for data acquisition. The filename has to be entered by the operator.

Mainline section

The program checks a flag to see if there were any parameters sent by the background program. If yes, the new parameters are put in the appropriate tables, and a new "receive parameter request" will be issued.

An ITYPE greater or equal than 20 is the signal that the operator wants to close the data acquisition file and open a new one.

The GC report, sent by the HP 1000 GC computer, is read in subroutine GETBC

The timer waits for the next sample time.

The GC is triggered (by a digital output signal) to get a new sample.

All other process variables are read by subroutine DATAC.

Control settings are calculated in subroutine CONTR.

The signals to the valve controllers are sent by subroutine PUT.

Every sample time, the most important data are being sent to disk file.

If the background program is running, data will be sent for operator presentation.

NAME: DOUT LOCATION: DOUT.MAC

DATE: 800114 AUTHOR: R.L. BARTON

ABSTRACT: Digital output driver, to set the digital outputs on the LSI-11 to the desired values.

CALLING SEQUENCE: CALL DOUT(IDUM,IDVEC)

where:

IDUM....not used

IDVEC = .BYTE - not used
        .BYTE - module number, use 0
        .WORD - data word

To set switches 2 and 6 to "on", the data word would be:

0000000000100010

CALLED BY: FOR2.....Fortran program
           PUT......Fortran Subroutine

CALLS:

DESCRIPTION: See program listing

NAME: GLUE LOCATION: ASSBA.MAC

DATE: 791112 AUTHOR: G.W.M. Coppus

ABSTRACT: Subroutine takes two real values and puts them in a four word integer array.

CALLING SEQUENCE: CALL GLUE(REAL1,REAL2,IARRAY)

where:

REAL1, REAL2 are two real numbers

IARRAY is a four word integer buffer

CALLED BY: FOR2.....Fortran program

CALLS:

NAME: GETBC LOCATION: SUBF2.FOR

DATE: 800218 AUTHOR: G.W.M. Coppus

ABSTRACT: Subroutine reads GC report from HP 1000 computer (LARRY) and gets the value for the bottom composition as an integer number.

CALLING SEQUENCE: CALL GETBC

CALLED BY: FOR2.....Fortran program

CALLS: SECNDS.....System Subroutine
MTIN.......System Subroutine
INDEX......System Subroutine
ASCINT.....Assembler Subroutine

DESCRIPTION:

First, the subroutine calculates the time available for finding the value of the bottom composition. This time is:

TREP = SAMPLE TIME - GC-DELAY - 2 = 180 - 140 - 2 = 38 seconds

Then the subroutine starts looking for characters on the GC channel, while continually checking the elapsed time, until:

1) The time TREP has expired. Then the subroutine will exit whether it has found a new value for the bottom composition or not. This is done to avoid a program lock due to GC failure.
2) A carriage return character (octal 15) is

encountered. Then the program assumes that one line of input is complete and the character buffer will be searched for the character string &WATER.

3) The character buffer is full, i.e. there are 40 characters without a carriage return. From the GC report we know that the characterstring &WATER should occur within the first 40 characters of a line. The character buffer will be searched for the character string.

If we find the string &WATER, we know that the 6 ASCII characters preceding the string represent the weight percentage of water in the sample, e.g. 94.556 &WATER. From this we can deduct that the methanol concentration is 5.444 %wt.

The assembler routine ASCINT takes care of the conversion from ASCII to integer.

Whether the bottom product composition is found or not, the subroutine will jump back in the loop (see above) to look for new characters until the time TREP has expired.

The reasons for looking for the water composition and not the methanol composition are:

1) Sometimes the word "METHANOL" is missing from the GC report, "&WATER" seems to be always there, so this is more reliable.

2) Sometimes, when the sample valve does not operate properly, the GC analysis shows three peaks instead of two. One of the smaller peaks will be identified as the "methanol peak", the large peak is the "water peak". We find that 100% - water concentration is in this case a more accurate representation of the methanol concentration than the number that was found from the peak that was identified as the methanol peak, so we get increase accuracy.

It should be emphasized that a single erroneous GC reading will have disastrous results for a run since this reading will cause a big jump in the steam flowrate , the effects of which take about one hour to die out.

NAME: TIMER LOCATION: SUBF2.FOR

DATE: 800125 AUTHOR: G.W.M. Coppus

ABSTRACT: Subroutine calculates the time for the next sample moment and suspends program execution until then.

CALLING SEQUENCE: CALL TIMER

CALLED BY: FOR2.....Fortran program

CALLS: JICVT....System Subroutine
JJCVT....System Subroutine
JADD.....System Subroutine
CVTTIM...System Subroutine
IUNTIL...System Subroutine

DESCRIPTION:

The beginning of the subroutine is straight forward, the new sample time is calculated by adding the sample interval ISEC to the previous time. If, however, the calculated time is 24 hours, obviously 0 hours (midnight) is meant, since the internal clock of the LSI-11 switches to 00:00 after 23:59. If for some reason the calculated time is later than the present time, the program will continue immediately, without waiting.

NAME: DATAC LOCATION: SUBF2.FOR

DATE: 791114 AUTHOR: G.W.M. Coppus

ABSTRACT: Subroutine reads all analog inputs

CALLING SEQUENCE: CALL DATAC

CALLED BY: FOR2.....Fortran program

CALLS: AIHL.....Assembler Subroutine
IANAL....Assembler Subroutine
IPEEKB...System Subroutine

DESCRIPTION:

Starting at the first IO vector in the I/O-table, the subroutine checks if the vector is used (unused vectors start with the characters UN).

If yes, the info word in the IO vector (see section on buffer usage) will be analyzed. If the vector represents an analog input, the appropriate channel will be read.

Repeat above procedure until all vectors (12) are done.

NAME: CONTR LOCATION: CONTR.FOR

DATE: 791114 AUTHOR: G.W.M. Coppus

ABSTRACT: Control algorithm, calculates P, I, D, P-Robust and Feed forward control action.

CALLING SEQUENCE: CALL CONTR(ERR,SUM,Y)

where:

ERR = values of error, YREF-Y

SUM = values of error integral for I action

Y = values of controlled outputs (top and bottom compositions)

CALLED BY: FOR2.....Fortran program

CALLS: ENG.....Fortran function

INGE.....Fortran subroutine

DESCRIPTION:

This subroutine offers a variety of control algorithms, the algoritm is chosen by selecting ITYPE. The meaning of ITYPE is:

ITYPE = 0 no control action

1 proportional control	2 = P+feed forward
3 PI-control	4 = -+feed forward
5 PD-Robust control	6 = -+feed forward
7 PID-control	8 = -+feed forward
9 P-Robust control	10 = -+feed forward
11 Set steady state values	

First the program checks if ITYPE = 11. If yes, the last output values for feed, reflux and steam will replace the old steady state values. (The default values are: Feed 18 g/s, Reflux 9.5 g/s, Steam 14 g/s). Then ITYPE will be set to 0, which means no control (manual). If ITYPE is in the range 1 - 10, the control subroutine will get the feed flowrate, top product composition and bottom product composition in engineering units. Then the error vector will be calculated (YREF-Y). A "computed GO TO" selects the next program part. The integral part and P-robust part have anti reset windup which sets the value of the integral action to zero if it exceeds a certain value. Feed forward action will be included for any even value of ITYPE.

NAME: PUT LOCATION: SUBF2.FOR

DATE: 791114 AUTHOR: G.W.M. Coppus

ABSTRACT: Subroutine sends all output signals to Current Output Stations.

CALLING SEQUENCE: CALL PUT

CALLED BY: FOR2.....Fortran program

CALLS: IANAL....Assembler Subroutine
DOUT.....Assembler Subroutine
AOHL.....Assembler Subroutine
ISLEEP...System Subroutine

DESCRIPTION:

This subroutine sets the Current Output Stations (COS) to the calculated values. There is *one* analog output connected from the LSI-11 to the COS. The proper COS is selected by digital outputs, i.e. closing switch number 1 on the digital output card at the LSI-11 will cause COS #1 to respond to the analog output voltage from the LSI-11.

The subroutine checks every IO vector in the I/O-table. If the vector is used and an analog output, first all digital outputs will be set to zero to blank out the switching operation in the LSI, then the analog output will be set to the calculated voltage and the appropriate digital output will be set. Between the various switching operations in the subroutine there are wait statements (wait 0.1 secs) to enable the slow switches and COS to follow the outputs.

NAME: COMP LOCATION: SUBF2.FOR

DATE: 791112 AUTHOR: G.W.M. Coppus

ABSTRACT: Completion subroutine for parameter transmission.

Sets a flag.

CALLING SEQUENCE: parameter in CALL IRCVDF(.,.,.,COMP)

CALLED BY: IRCVDF...System Subroutine

CALLS:

DESCRIPTION:

All this subroutine does is setting FLAGP=.TRUE.

NAME: COMD LOCATION: SUBF2.FOR

DATE: 791112 AUTHOR: G.W.M. Coppus

ABSTRACT: Completion subroutine for data transmission. Sets a flag

CALLING SEQUENCE: parameter in CALL ISDATF(.,.,.,.,COMD)

CALLED BY: ISDATF...System Subroutine

CALLS:

DESCRIPTION:

All this subroutine does is setting FLAGD=.TRUE.

NAME: ASCINT LOCATION: CONV.MAC

DATE: 791112 AUTHOR: G.W.M. Coppus

ABSTRACT: Converts 5 ASCII characters into an integer number

CALLING SEQUENCE: CALL ASCINT(IBOT,IBCOMP)

where:

IBOT is a 5 byte logical array containing 5 ASCII digits

IBCOMP is an integer number 0 - 30000

CALLED BY: GETBC....Fortran Subroutine

CALLS:

DESCRIPTION:

This subroutine takes an array of characters representing the weight % of water in a Gas Chromatograph sample, e.g. 97555 means 97.555 %wt water. This number will be converted to the integer 27555. By substracting this from 30000 we get the %wt methanol in the sample. An identical conversion (to integer 97555) is not possible because integer numbers only go up to 32674. See description of GETBC for more explanation.

NAME: IANAL LOCATION: ASSBA.MAC

DATE: 791112 AUTHOR: G.W.M. Coppus

ABSTRACT: Subroutine analyzes the information bits and channel number byte in word 4 of an IO vector.

CALLING SEQUENCE: CALL IANAL(IOTAB(4,I),IO,IAD,ISQ,N)

where:

IOTAB(4,I) = info word in IO vector number I

on return from the subroutine:

IO = 1 for an input
     0 for an output

IAD = 1 for analog
      0 for digital

ISQ = 0 for linear
      1 for square
      2 for square root
      3 for temperature

N = channel number 0 - 256

CALLED BY: DATAC....Fortran Subroutine
ENG.....Fortran Function
INGE.....Fortran Subroutine
PUT......Fortran Subroutine

CALLS:

DESCRIPTION:

Analyzes the info word in the IO vector bit by bit. See the description of IO vectors in the section on usage of buffers.

NAME: AIHL LOCATION: AIHL.MAC

DATE: 790831 AUTHOR: R.L. Barton

ABSTRACT: Subroutine reads an analog input on the indicated channel and returns the voltage as an integer number representing millivolts.

CALLING SEQUENCE: CALL AIHL(IDUM,IDVEC)

where:

```
IDUM     = dummy
IDVEC(1) = unused (set to zero)
IDVEC(2) = channel #, 0 - 63 on entry,
                     decimal 4 on return
IDVEC(3) = number of bytes in record (8) on return
IDVEC(4) = address of data buffer (DATABF) on return
   DATABF = .BYTE count byte
            .BYTE Link Level Control
            .BYTE Message Level Control
            .BYTE low data byte
            .BYTE high data byte
            .BYTE not used
            .BYTE error code
```

CALLED BY: DATAC....Fortran Subroutine

CALLS:

DESCRIPTION:

For a detailed description, see the program listing. On return from the call, bytes 4 and 5 in the data buffer

(DATABF) contain the integer number which represents the voltage reading. See DATAC to find out how to read from the data buffer.

NAME: ENG LOCATION: SUBF2.FOR

DATE: 800218 AUTHOR: G.W.M. Coppus

ABSTRACT: Subroutine takes an IO vector as input and returns the process value in engineering units.

CALLING SEQUENCE: VALUE = ENG(IOTAB(1,I)

where:

VALUE = number in engineering units

IOTAB(1,I) = IO vector from the I/O-table.

CALLED BY: CONTR....Fortran Subroutine

CALLS: SPLIT....Assembler Subroutine
IANAL....Assembler Subroutine

DESCRIPTION:

Depending on the type of input (linear or square root as in flows), and the values for SPAN and ZERO, the function will calculate the value in engineering units using the algorithm:

```
IVAL is the integer representing voltage (mVolts)
RVAL = (IVAL-1000)/4000 for an input
       IVAL/10000 for an output
CVAL = RVAL**0.5, 1 or 2 for SQRT, LIN, SQ
 ENG = SPAN*CVAL + ZERO
```

NAME: INGE LOCATION: SUBF2.FOR

DATE: 791115 AUTHOR: G.W.M. Coppus

ABSTRACT: Calculates the integer value from the number in engineering units and puts the number in the IO vector. Opposite of ENG.

CALLING SEQUENCE: CALL INGE(X,IOTAB1,I)

where:

X = value in engineering units
IOTAB(1,I) = IO vector where integer is to be put into

CALLED BY: CONTR....Fortran Subroutine

CALLS: SPLIT....Assembler Subroutine
IANAL....Assembler Subroutine

DESCRIPTION:

The algorithm is:

```
   CVAL = (X-ZERO)/SPAN
   RVAL = CVAL for a linear output
          CVAL**0.5 for a squared output
          CVAL**2 for a square root output
IVEC(3) = 10000*RVAL
```

NAME: SPLIT LOCATION: ASSBA.MAC

DATE: 791112 AUTHOR: G.W.M. Coppus

ABSTRACT: Splits 4 integer words into 2 real values

CALLING SEQUENCE: CALL SPLIT(IARRAY,REAL1,REAL2)

where:

IARRAY = four word integer array

REAL1 = first real value

REAL2 = second real value

CALLED BY: ENG......Fortran Function

CALLS:

NAME: AOHL LOCATION: AOHL.MAC

DATE: 791212 AUTHOR: R.L. Barton

ABSTRACT: Analog output driver, sends an analog output to the indicated channel.

CALLING SEQUENCE: CALL AOHL(IDUM,IDVEC)

where:

```
IDVEC = .BYTE not used
        .BYTE module number (0)
        .BYTE converter on module 0-3, we use 0
        .BYTE not used
        .WORD integer +10000 - -10000 (mVolts)
```

CALLED BY: PUT......Fortran Subroutine

CALLS:

DESCRIPTION:

See the program listing for a detailed description. The way the control system is set up, we use module #0, converter #0 as the analog output channel to drive the Current Output Stations.

NAME: BACK3 LOCATION: BACK3.FOR

DATE: 791112 AUTHOR: G.W.M. Coppus

ABSTRACT: Mainline background program, the operator interface.

CALLING SEQUENCE:

CALLED BY:

CALLS: ASSIGN...System Subroutine
CLOSE....System Subroutine
GLUE.....Assembler Subroutine
IRCVDF...System Subroutine
SPRINT...Fortran Subroutine
SERVE....Fortran Subroutine

DESCRIPTION:

In the initialization section, all initial values for parameters will be read from file INIT.DAT. Also, the event file (EVENT.DAT) will be opened and the time for the first scheduled event will be read.

The main body of the program consists of a continuous loop around the subroutine SERVE to check for operator commands and service them. It also checks for scheduled events. Once every sample time, the program jumps out of this loop to print the just received data on the screen.

NAME: SPRINT LOCATION: SUBBA.FOR

DATE: 791110 AUTHOR: G.W.M. Coppus

ABSTRACT: subroutine takes care of printing of data on the console

CALLING SEQUENCE: CALL SPRINT(PCODE)

where:

PCODE = 0 no printing
1 concise printing
2 I/O-table
3 CP-table
4 PP-table
5 all tables

CALLED BY: BACK3....Fortran program
SERVE....Fortran Subroutine

CALLS: CONC.....Fortran Subroutine
LISIO....Fortran Subroutine
LISCP....Fortran Subroutine
LISPP....Fortran Subroutine

DESCRIPTION:

Depending on the value of PCODE, the subroutine calls one or more of the appropriate listing routines.

NAME: CHANPP LOCATION: SUBBA.FOR

DATE: 791112 AUTHOR: G.W.M. Coppus

ABSTRACT: subroutine to enter changes in the program parameters table.

CALLING SEQUENCE: CALL CHANPP(IPAR,LU)

where:

IPAR = 2 integers indicating the parameter to change

LU = logical unit number to read new values from

CALLED BY: SERVE....Fortran Subroutine

CALLS:

DESCRIPTION:

This subroutine prompts the operator and then reads new parameter values into the PP-table.

NAME: CHANCP LOCATION: SUBBA.FOR

DATE: 791112 AUTHOR: G.W.M. Coppus

ABSTRACT: subroutine to enter changes in the controller parameters table.

CALLING SEQUENCE: CALL CHANCP(IPAR,LU)

where:

IPAR = integer number of parameter to change

LU = logical unit from where new values are to be read

CALLED BY: SERVE....Fortran Subroutine

CALLS:

NAME: CHANIO LOCATION: SUBBA.FOR

DATE: 791112 AUTHOR: G.W.M. Coppus

ABSTRACT: subroutine to enter changes in the controller parameters table.

CALLING SEQUENCE: CALL CHANIO(IPAR,LU)

where:

IPAR = two integers indicating the position in the table that has to be changed.

LU = logical unit number from where new values are to be read

CALLED BY: SERVE....Fortran Subroutine

CALLS: PUTIN....Fortran Subroutine

GLUE.....Assembler Subroutine

NAME: CONC LOCATION: SUBBA.FOR

DATE: 791112 AUTHOR: G.W.M. Coppus

ABSTRACT: subroutine lists a concise subset of data on the console

CALLING SEQUENCE: CALL CONC

CALLED BY: SPRINT...Fortran Subroutine

CALLS: ENG......Fortran Function

CVTTIM...System Subroutine

DESCRIPTION:

This subroutine writes the most important process variables on the console every sample time. At every fifth sample a line of mnemonics will be written.

NAME: LISPP LOCATION: SUBBA.FOR

DATE: 791112 AUTHOR: G.W.M. Coppus

ABSTRACT: subroutine lists the program parameters table on the console

CALLING SEQUENCE: CALL LISPP

CALLED BY: SPRINT...Fortran Subroutine

CALLS: CVTTIM...System Subroutine

NAME: LISCP LOCATION: SUBBA.FOR

DATE: 791112 AUTHOR: G.W.M. Coppus

ABSTRACT: subroutine lists the CP-table on the console

CALLING SEQUENCE: CALL LISCP

CALLED BY: SPRINT...Fortran Subroutine

CALLS:

NAME: LISIO LOCATION: SUBBA.FOR

DATE: 791112 AUTHOR: G.W.M. Coppus

ABSTRACT: subroutine lists the I/O-table on the console

CALLING SEQUENCE: CALL LISIO

CALLED BY: SPRINT...Fortran Subroutine

CALLS: SPLIT....Assembler Subroutine
IANAL....Assembler Subroutine

NAME: PUTIN LOCATION: SUBBA.FOR

DATE: 791112 AUTHOR: G.W.M. Coppus

ABSTRACT: subroutine puts a new bit pattern in the INFO word of an IO vector.

CALLING SEQUENCE: CALL PUTIN(IARG)

where:

IARG = word where bit pattern is to be put

CALLED BY: CHANIO...Fortran Subroutine

CALLS:

DESCRIPTION:

This subroutine expects the operator to enter a mnemonic for an IO vector. From this mnemonic the subroutine sets up the right bit pattern. E.g. If the operator wants to make a vector an output signal for a flow controller, the mnemonic would be SQRT AO (SQuare Root Analog Out) and the bit pattern would be 00001010. See description of buffer usage.

NAME: SERVE LOCATION: SUBBA.FOR

DATE: 791110 AUTHOR: G.W.M. Coppus

ABSTRACT: Subroutine to service operator commands and scheduled parameter changes.

CALLING SEQUENCE: CALL SERVE(LMPM)

where:

LMPM = time for next change in minutes past midnight

CALLED BY: BACK3....Fortran program

CALLS: SPRINT...Fortran Subroutine

SCHED....Fortran Subroutine

CHANPP...Fortran Subroutine

CHANCP...Fortran Subroutine

CHANIO...Fortran Subroutine

IPOKE....System Subroutine

IPEEK....System Subroutine

ITTINR...System Subroutine

ISDATF...System Subroutine

DESCRIPTION:

The operator can enter List and Change commands on the console. The subroutine analyzes the command and invokes the appropriate subroutine. Also, the subroutine checks the time of scheduled events, and reads the event from disk file at the right time for implementation.

NAME: SCHED LOCATION: SCHED.FOR

DATE: 791218 AUTHOR: G.W.M. Coppus

ABSTRACT: subroutine reads and implements events from an event file.

CALLING SEQUENCE: CALL SCHED(LMPM,SEND)

where:

LMPM = time for next event in minutes past midnight

SEND = logical flag to indicate a change

CALLED BY: SERVE....Fortran Subroutine

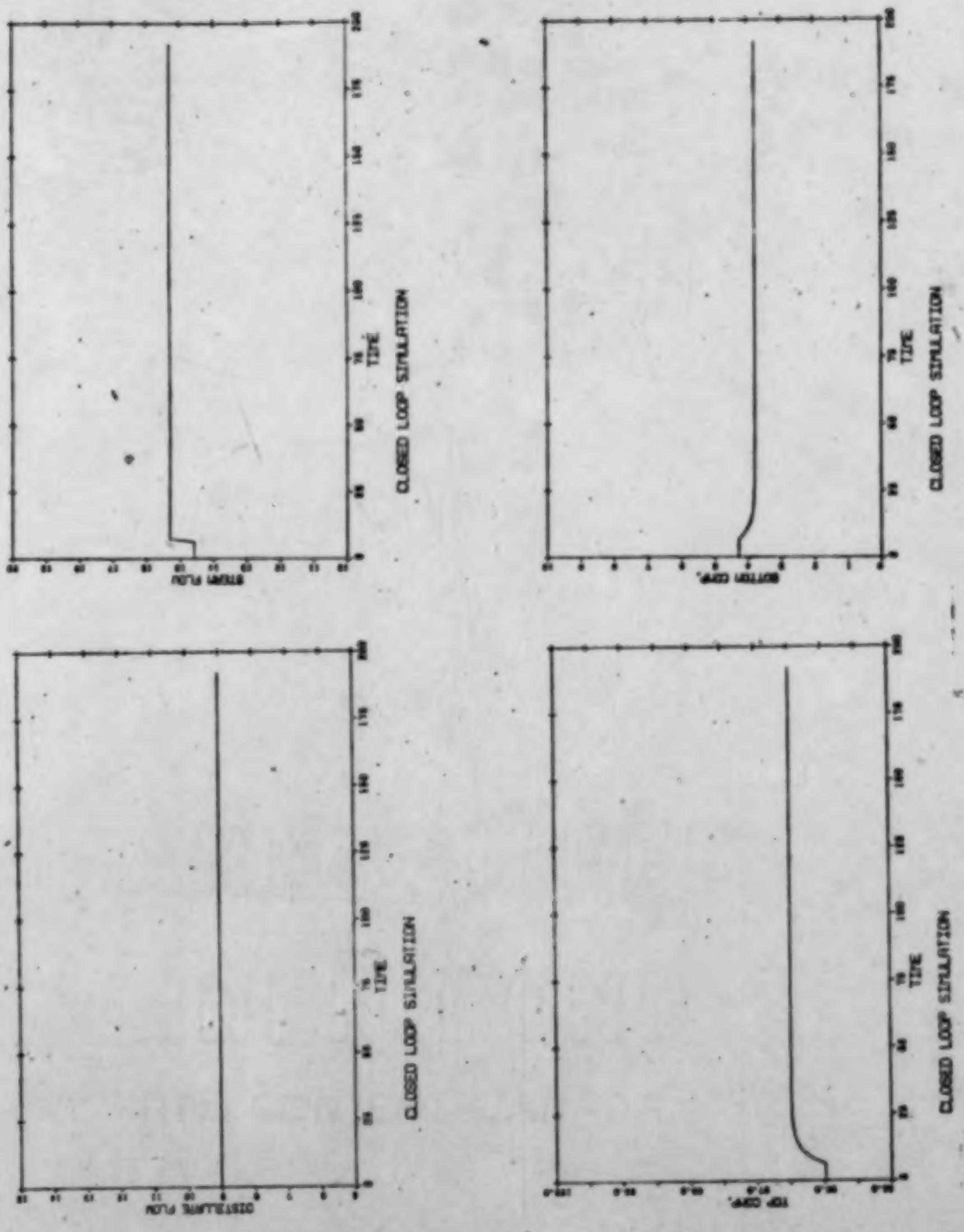
CALLS: CVTTIM...System Subroutine
CHANIO...Fortran Subroutine
CHANCP...Fortran Subroutine
CHANPP...Fortran Subroutine

DESCRIPTION:

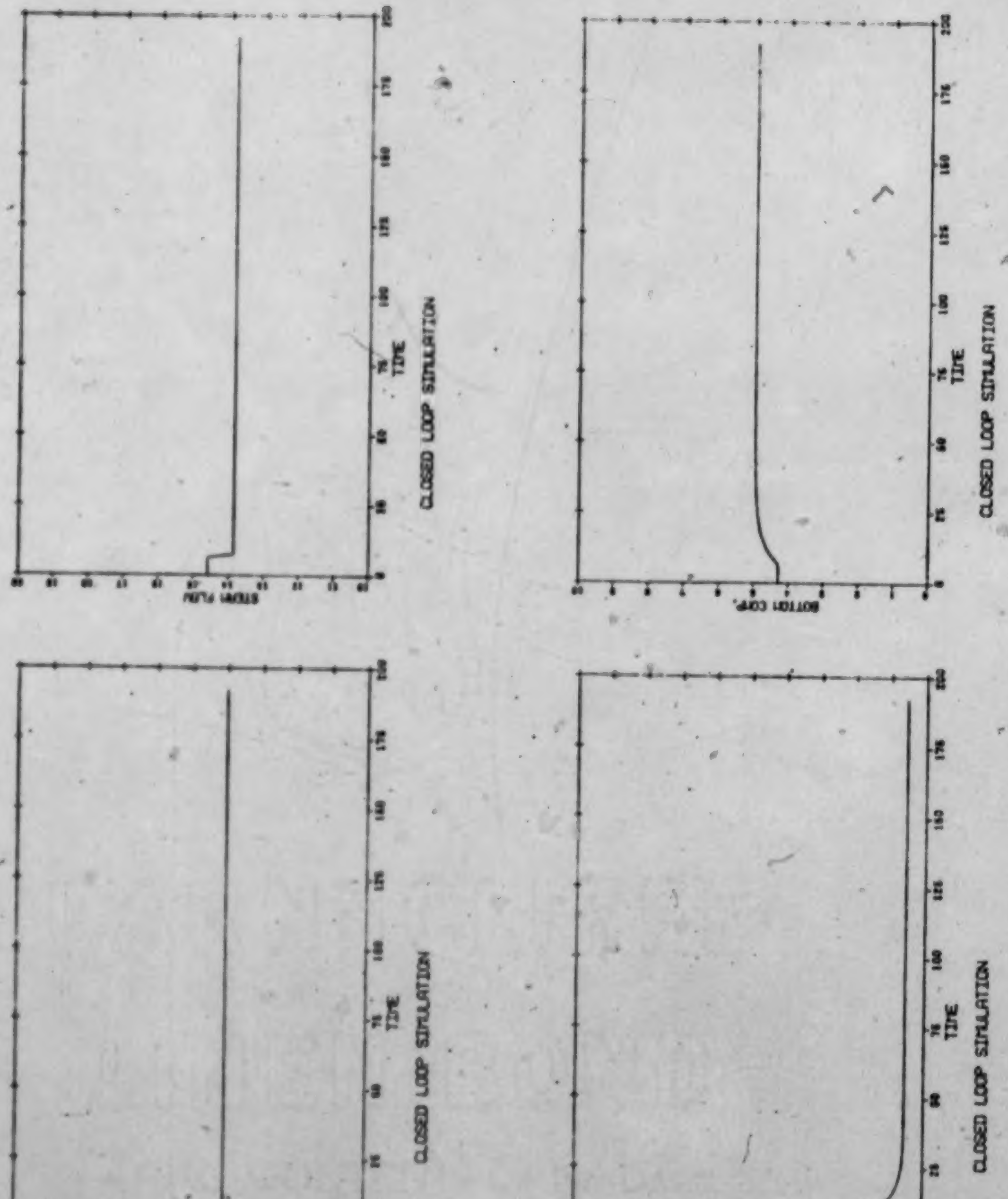
The subroutine checks the next sample time and compares it with the time for the next change. If the time for the next change is later than the next sample time, or not within ten minutes before the next sample time, nothing will happen.

Otherwise, the new parameters will be read from the event file and subroutine SERVE will be flagged to send them to the foreground program.

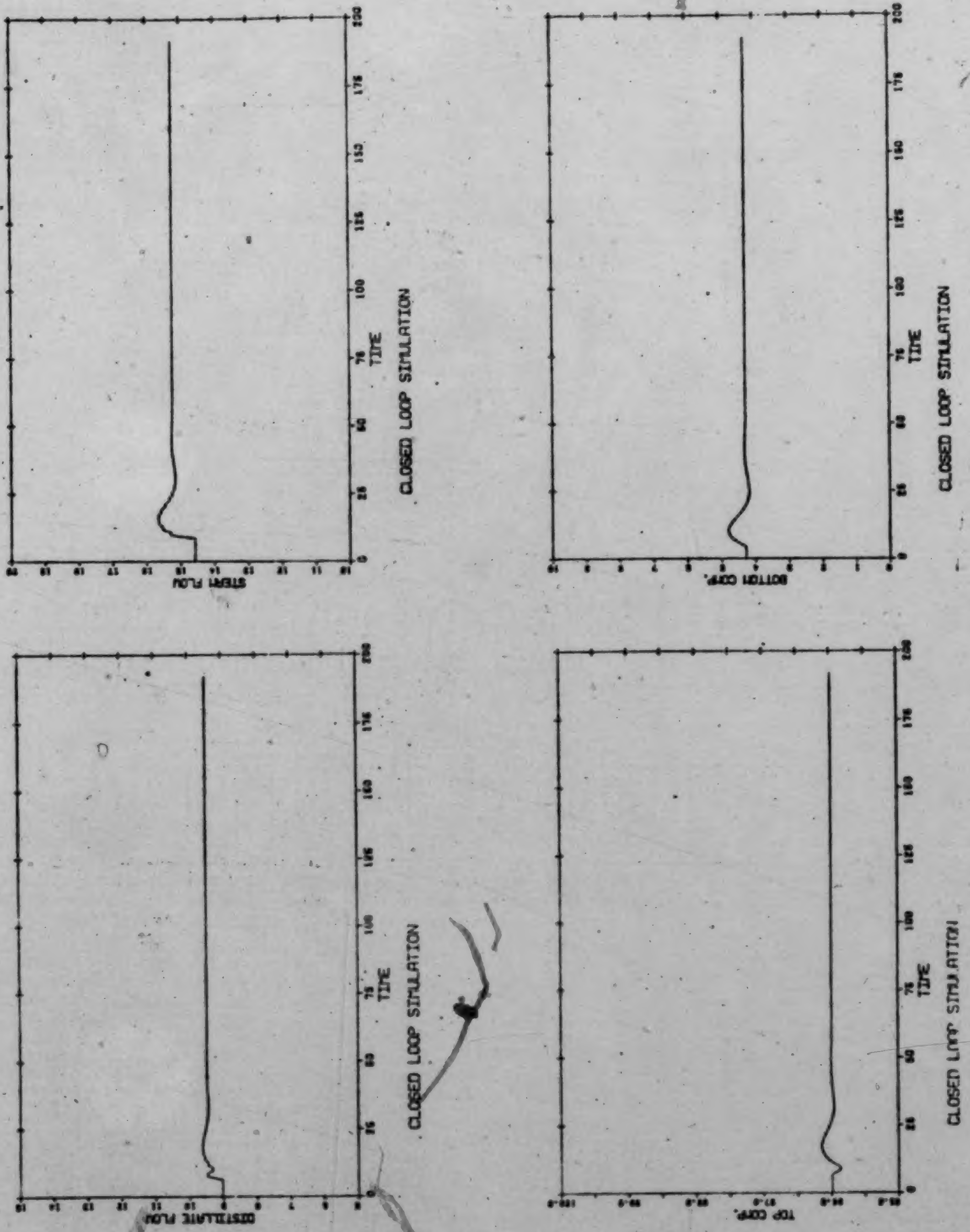
7. APPENDIX B: Plots of simulation and experimental responses



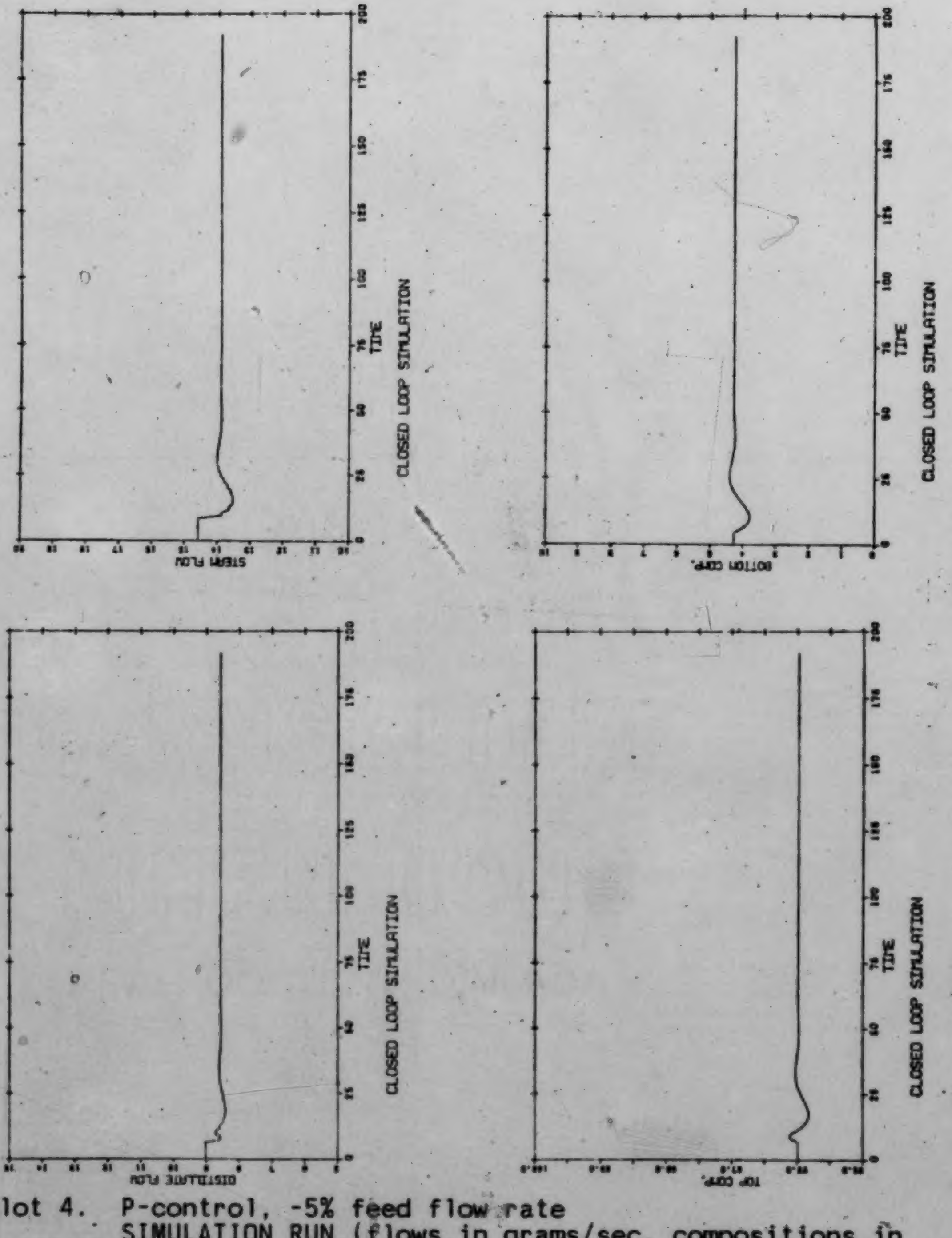
Plot 1. Open loop, +5% steam flow rate
SIMULATION RUN (flows in grams/sec, compositions in weight fraction methanol)



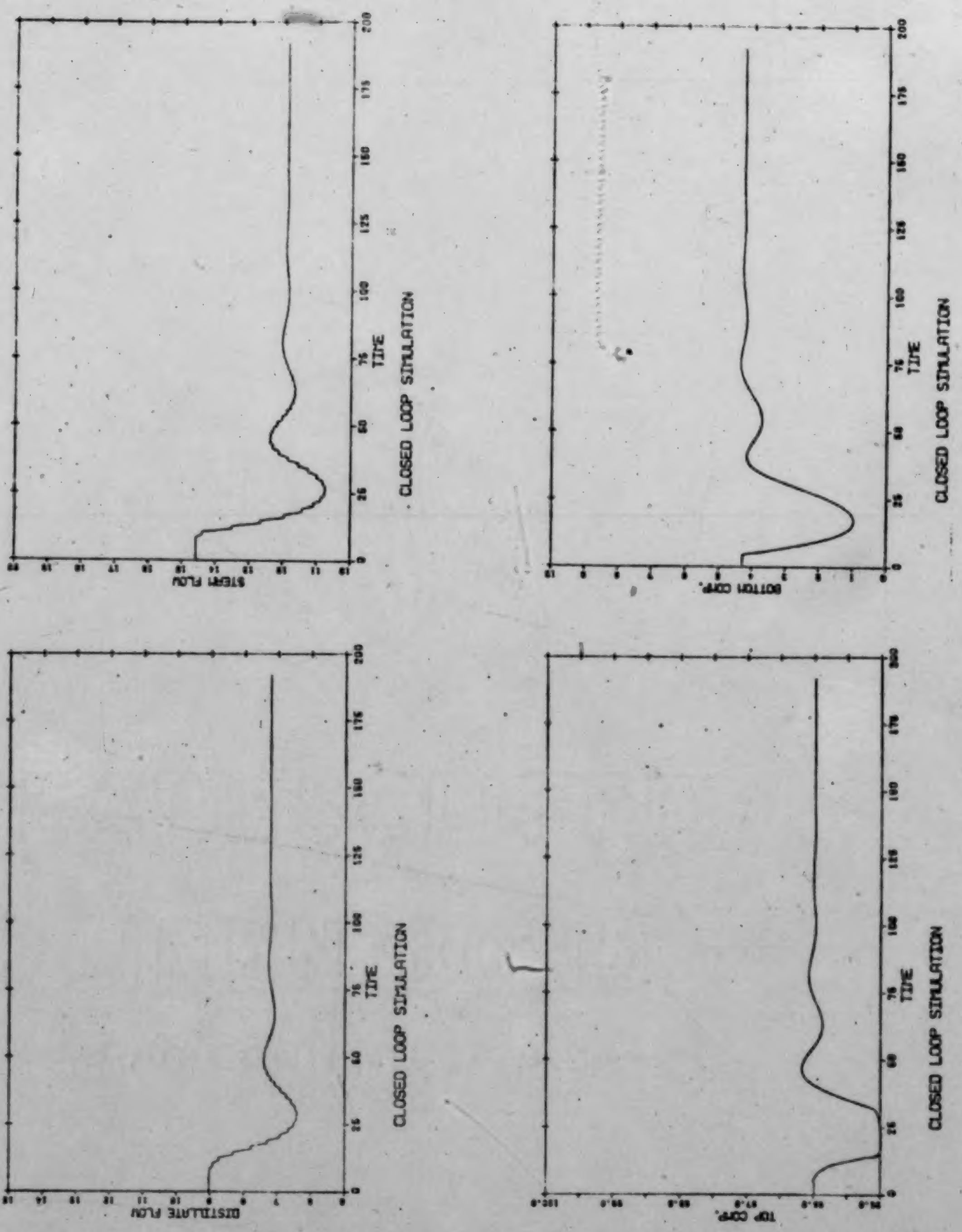
Plot 2. Open loop, -5% steam flow rate
SIMULATION RUN (flows in grams/sec, compositions in weight fraction methanol)



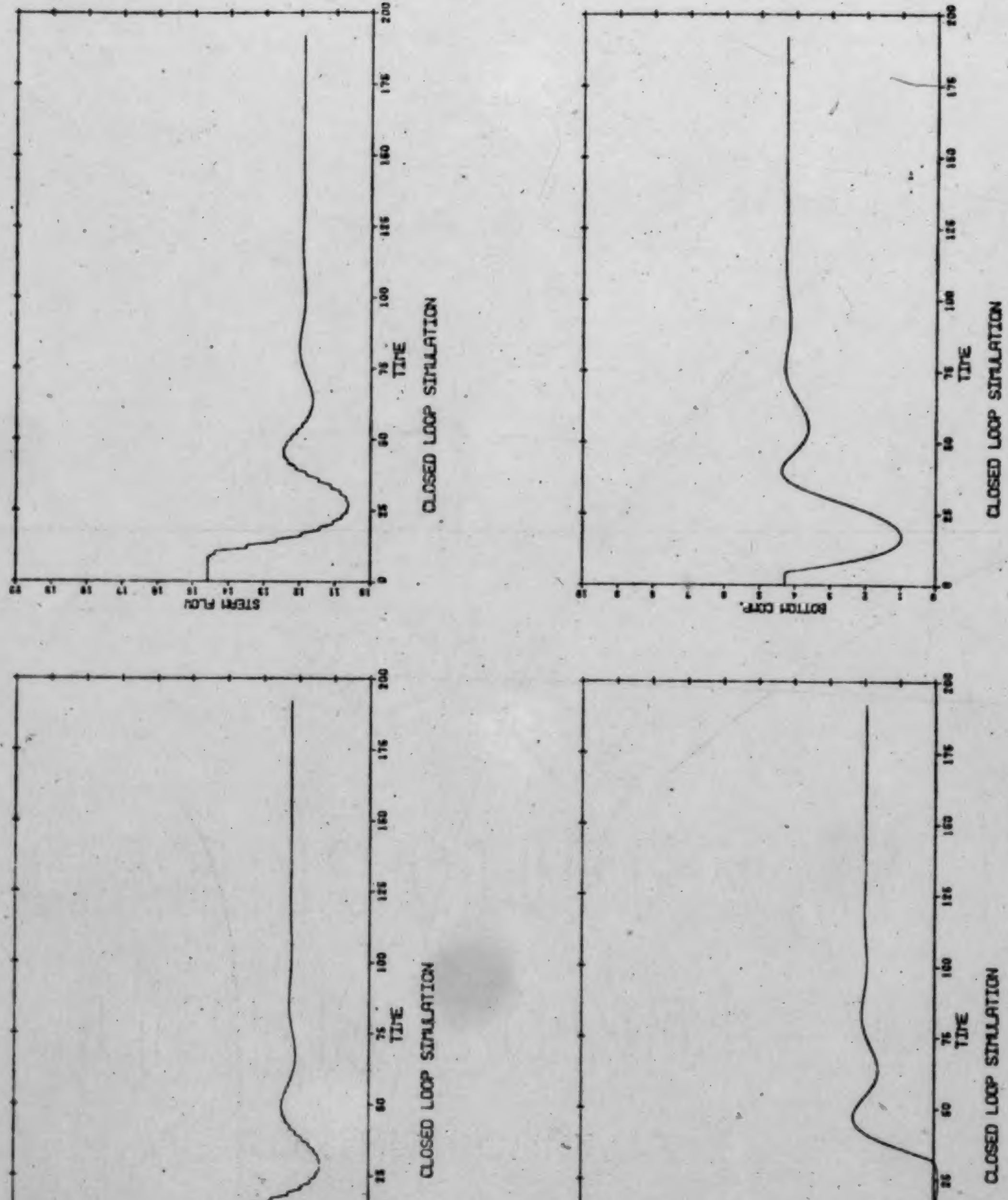
Plot 3. P-control, +5% feed flow rate
SIMULATION RUN (flows in grams/sec, compositions in weight fraction methanol)



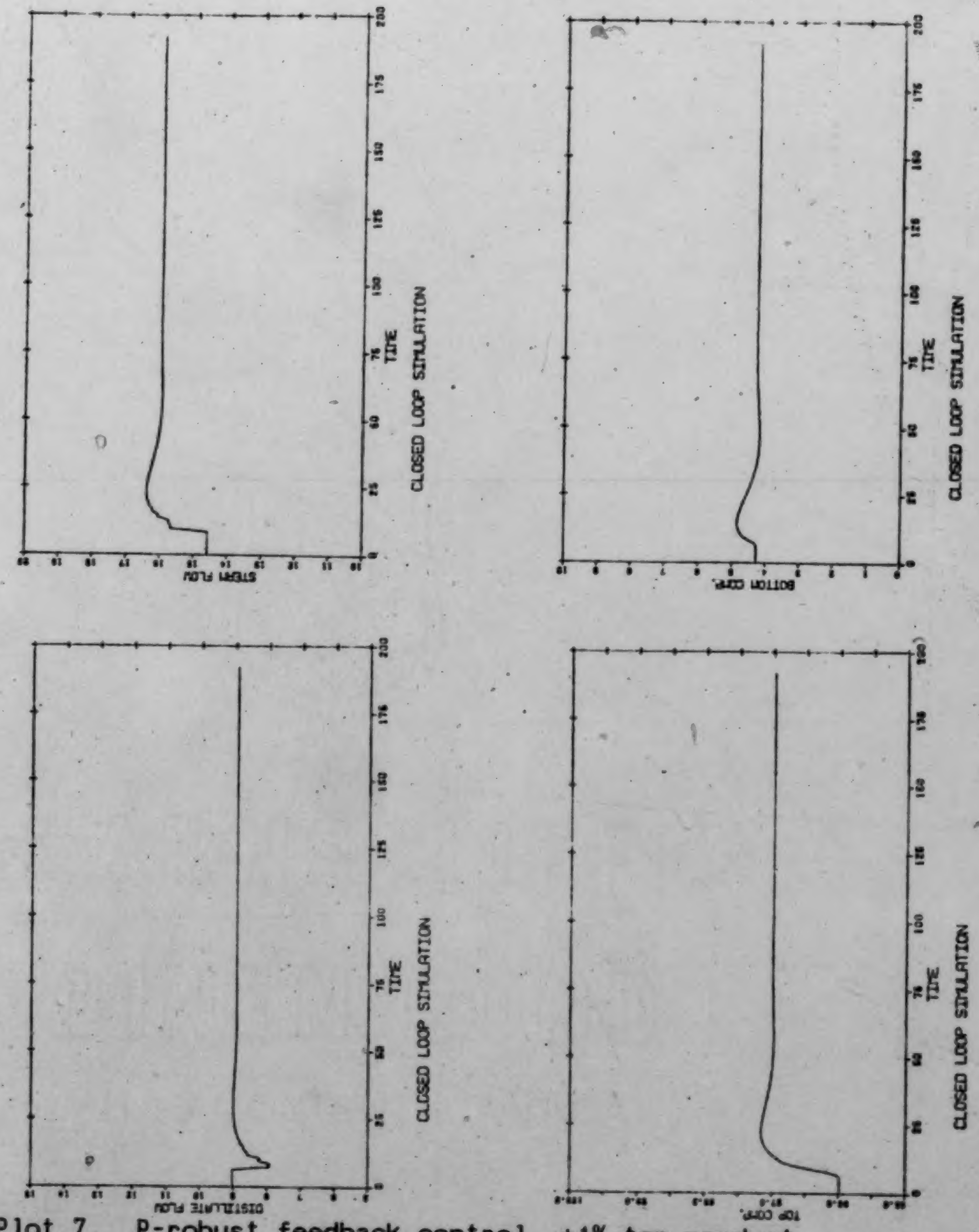
Plot 4. P-control, -5% feed flow rate
SIMULATION RUN (flows in grams/sec, compositions in weight fraction methanol)



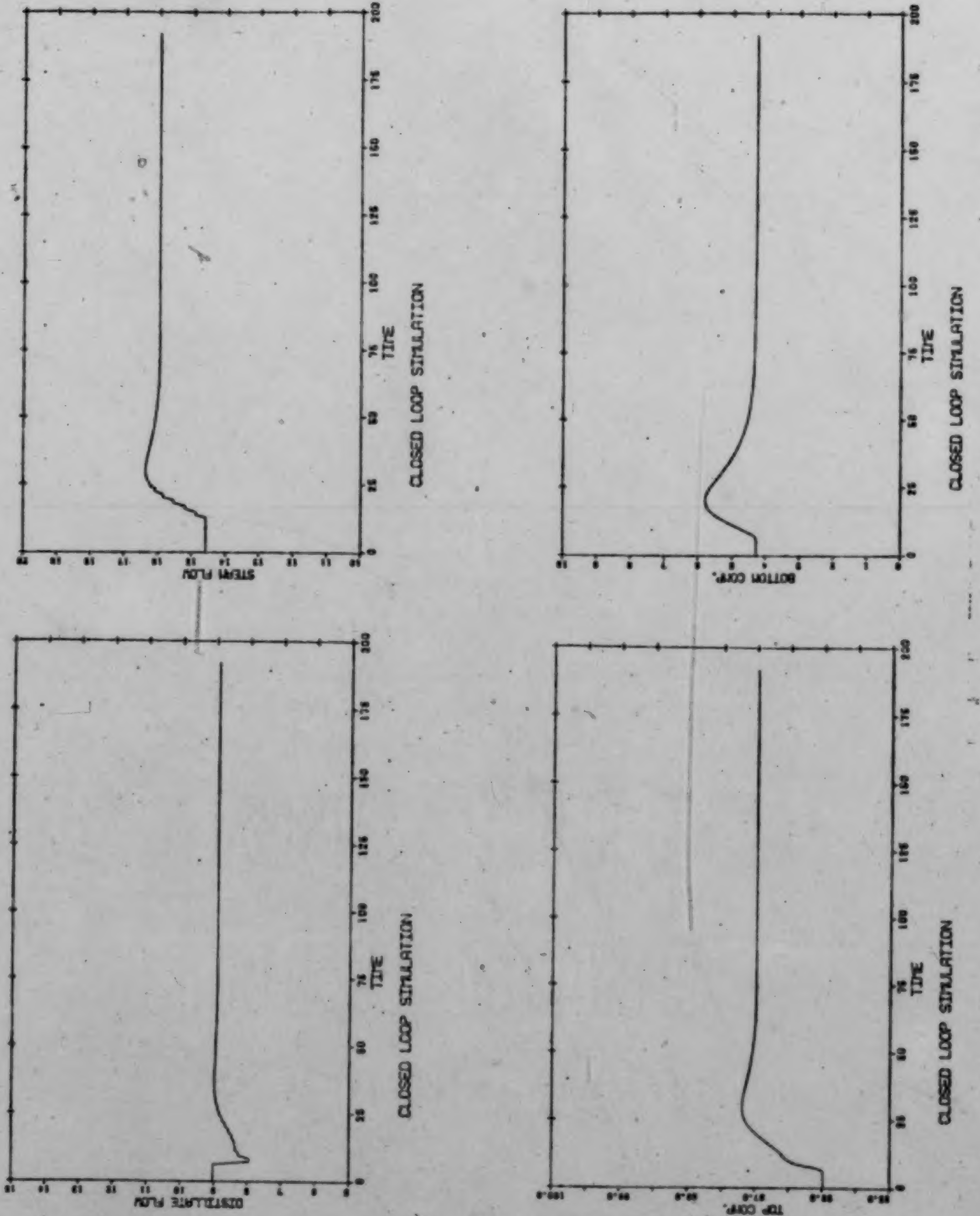
Plot 5. P-robust feedback control, -20% feed flow rate SIMULATION RUN (flows in grams/sec, compositions in weight fraction methanol)



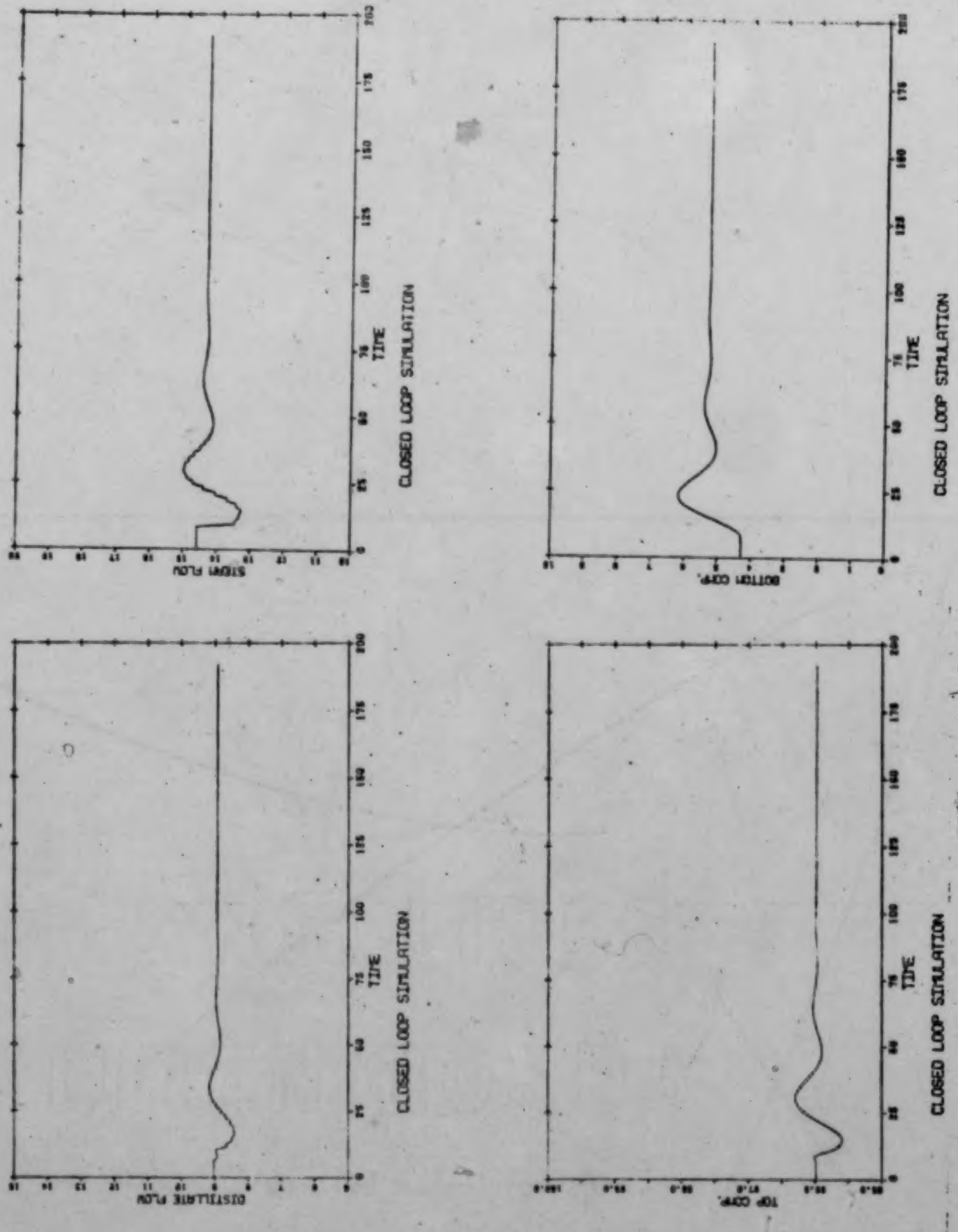
Plot 6. PI control, -20% feed flow rate
SIMULATION RUN (flows in grams/sec, compositions in weight fraction methanol)



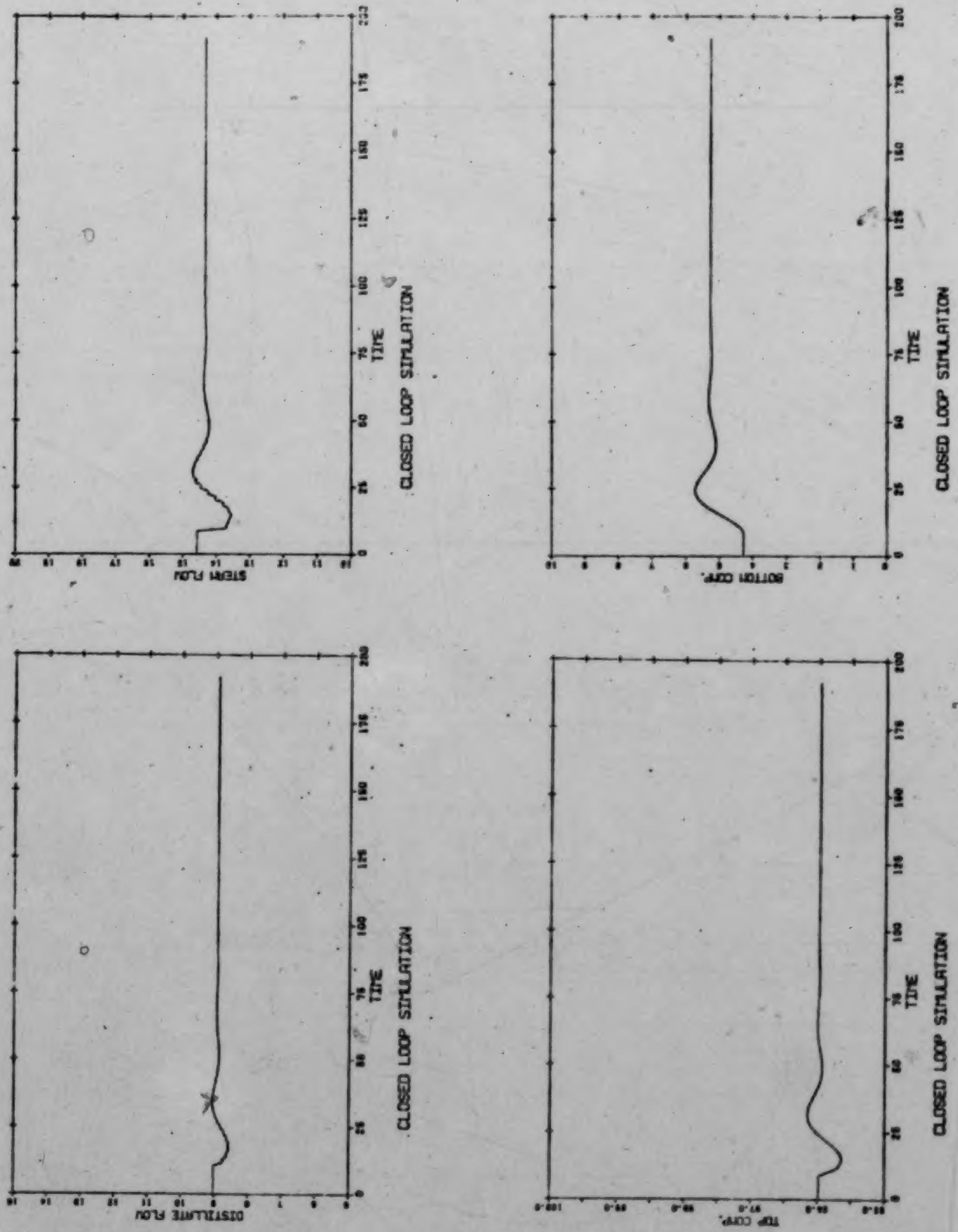
Plot 7. P-robust feedback control, +1% top product composition
SIMULATION RUN (flows in grams/sec, compositions in weight fraction methanol)



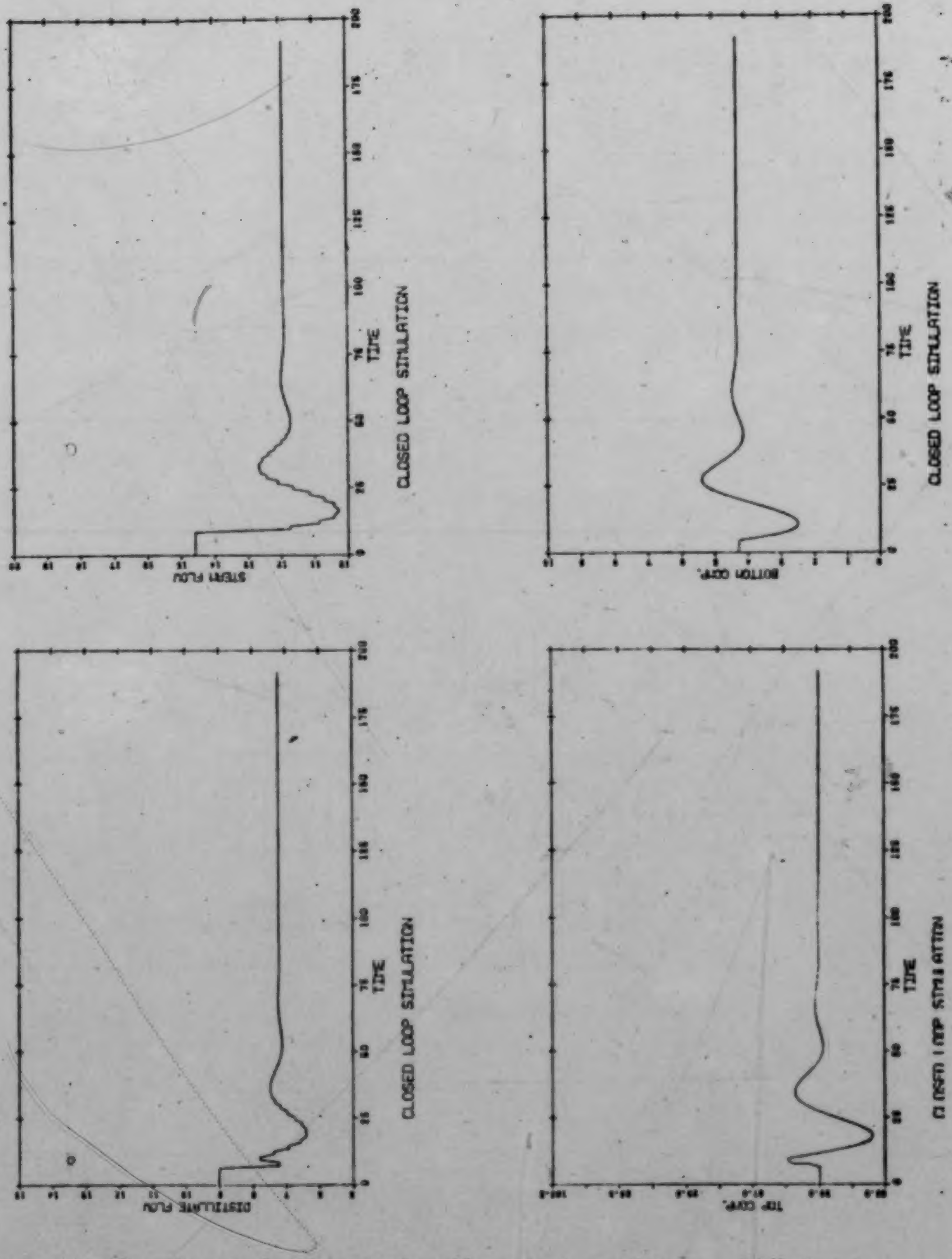
Plot 8. PI control, +1% top product composition SIMULATION RUN (flows in grams/sec, compositions in weight fraction methanol)



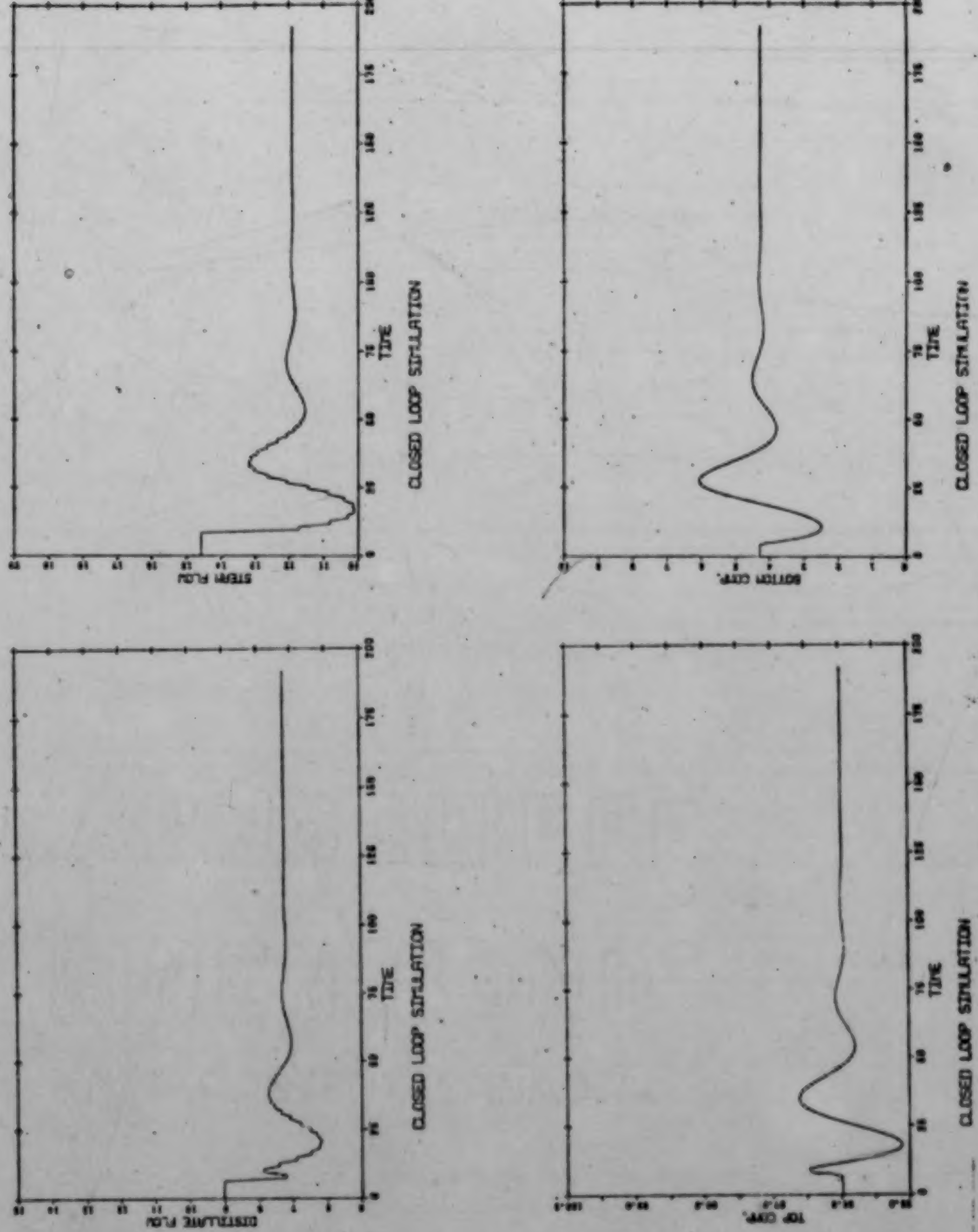
Plot 9. P-robust feedback control, +1% bottom product composition
SIMULATION RUN (flows in grams/sec, compositions in weight fraction methanol)



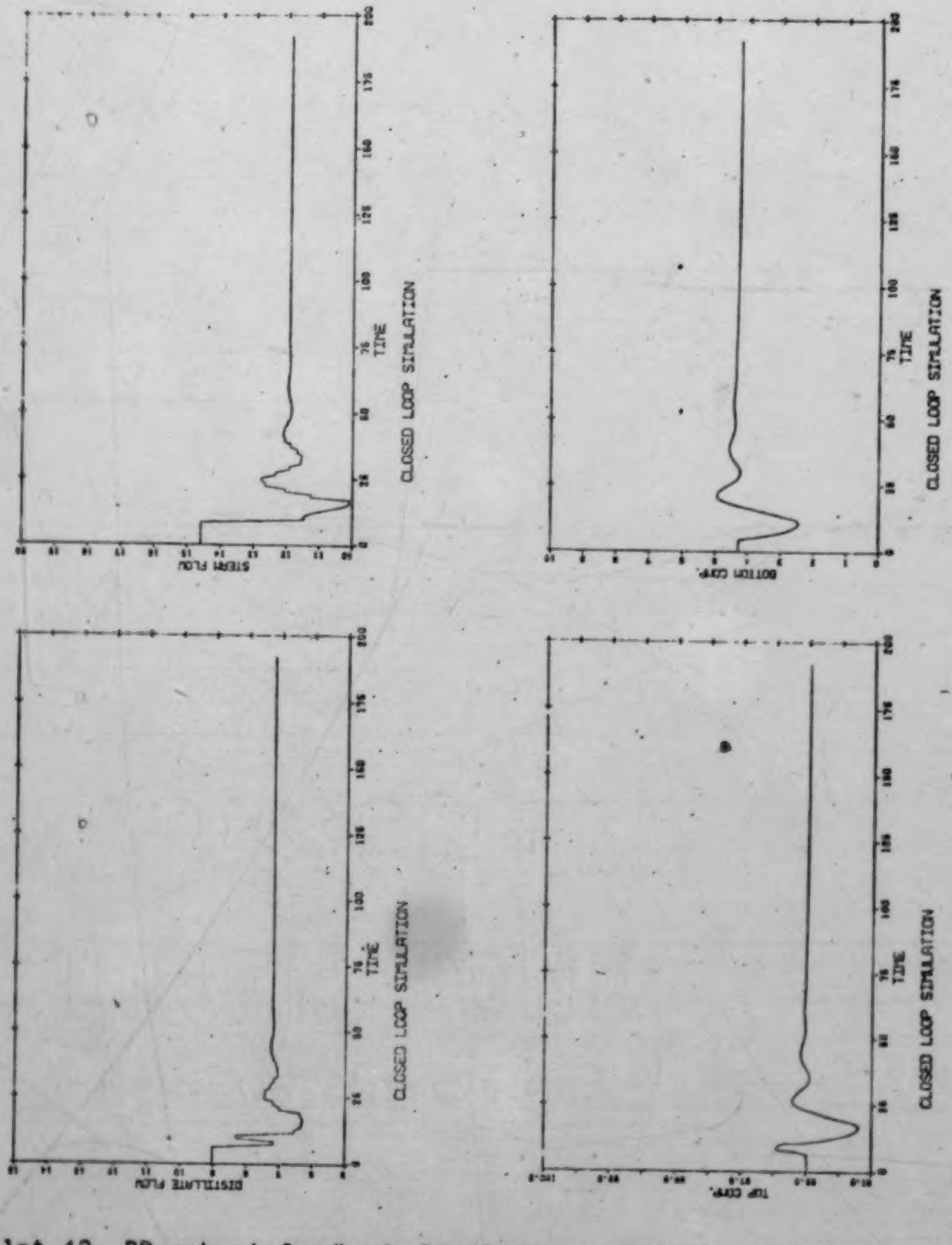
Plot 10. PI control, +1% bottom product composition SIMULATION RUN (flows in grams/sec, compositions in weight fraction methanol)



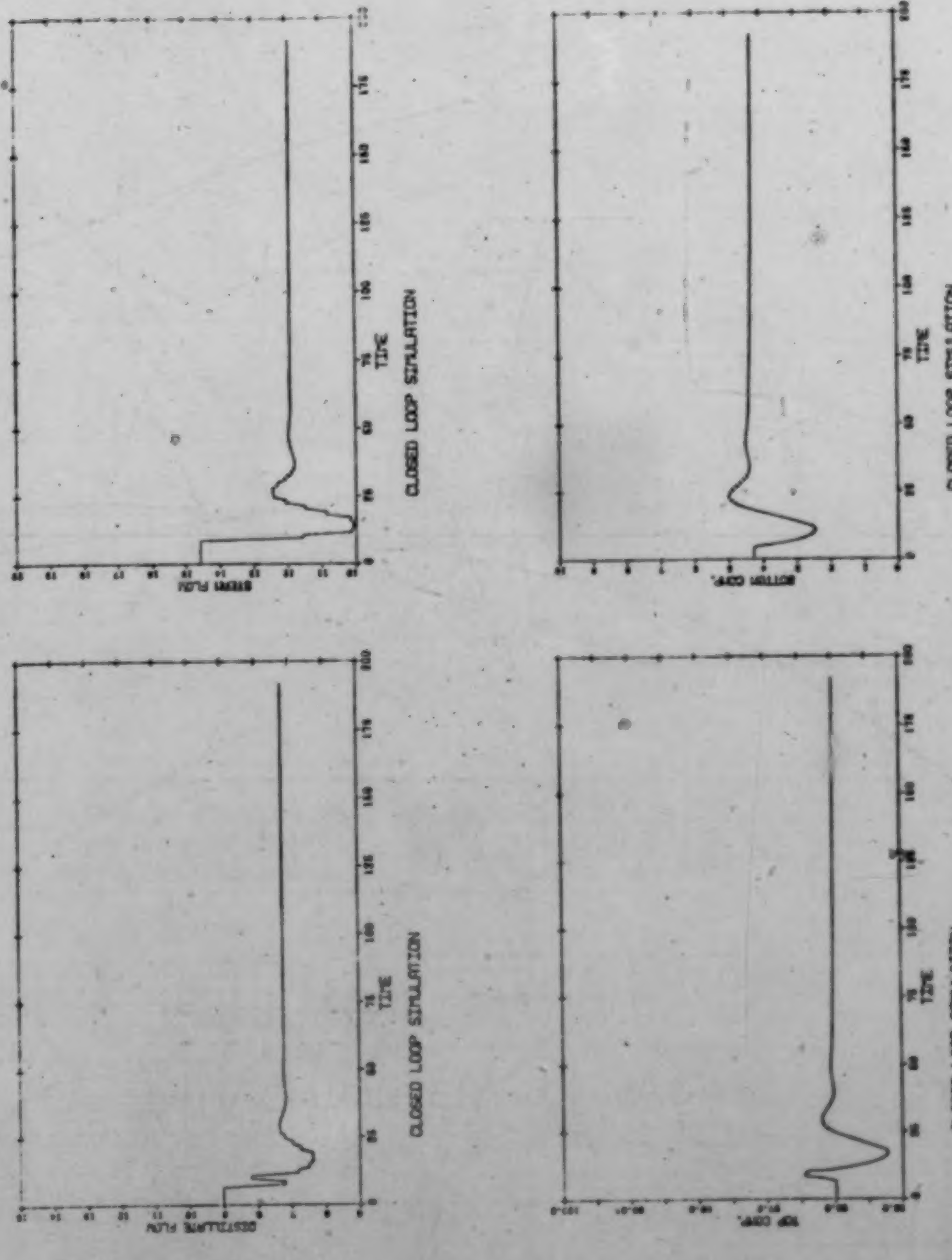
Plot 11. P-robust feedback feedforward control, -20% feed flow rate
SIMULATION RUN (flows in grams/sec, compositions in weight fraction methanol)



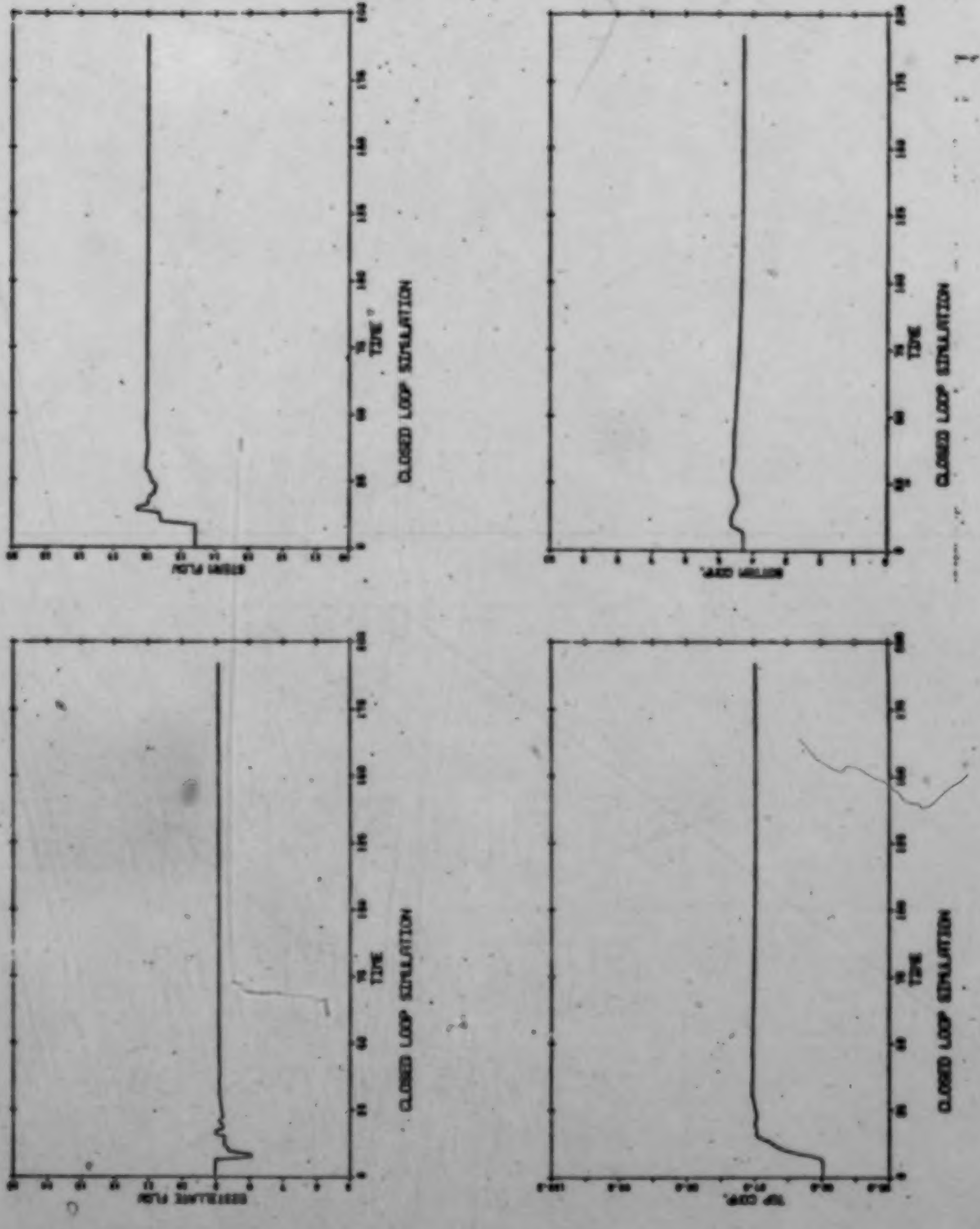
Plot 12. PI-plus-feedforward control, -20% feed flow rate SIMULATION RUN (flows in grams/sec, compositions in weight fraction methanol)



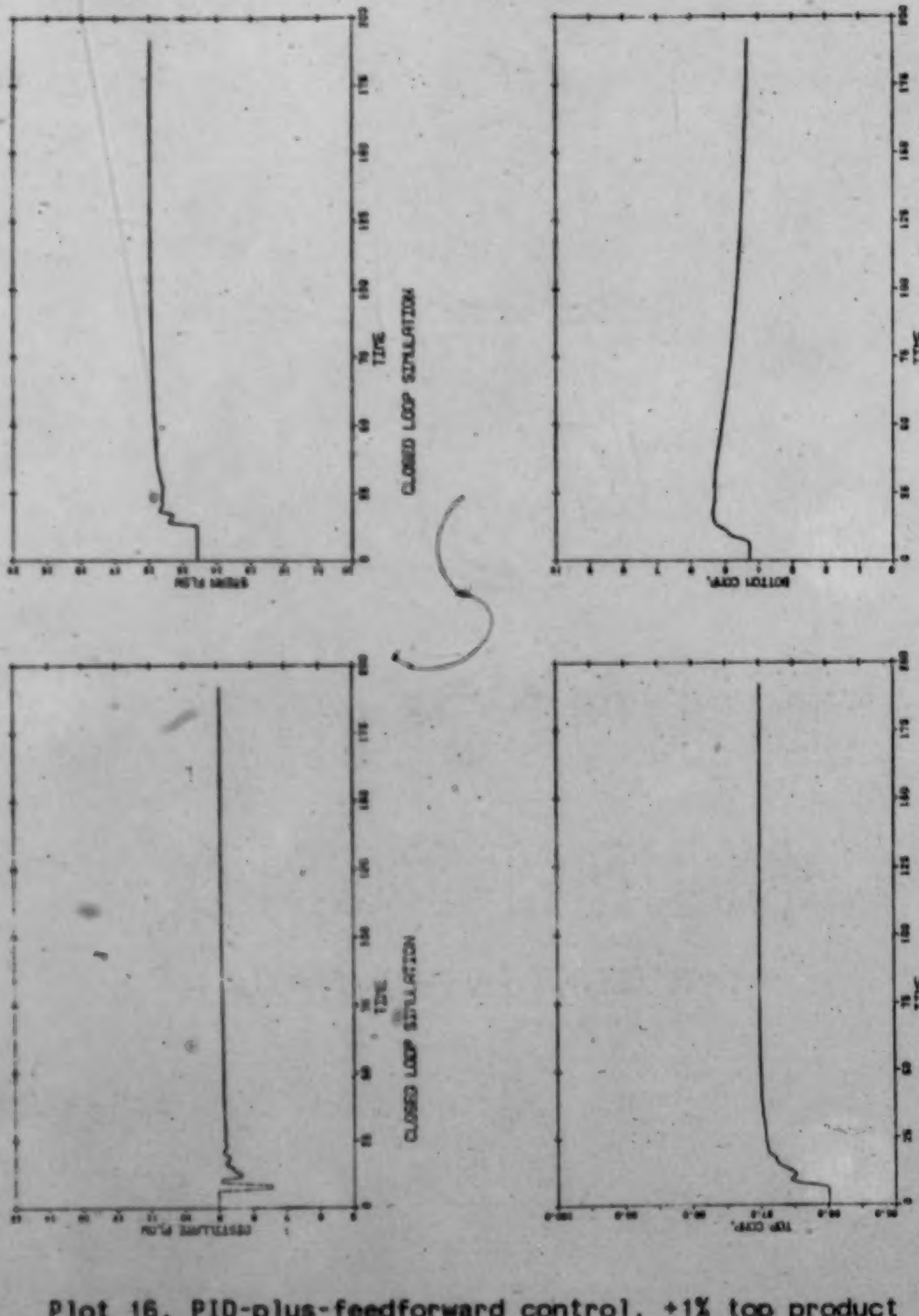
Plot 13. PD-robust feedback feedforward control, -20% feed flow rate
SIMULATION RUN (flows in grams/sec, compositions in weight fraction methanol)



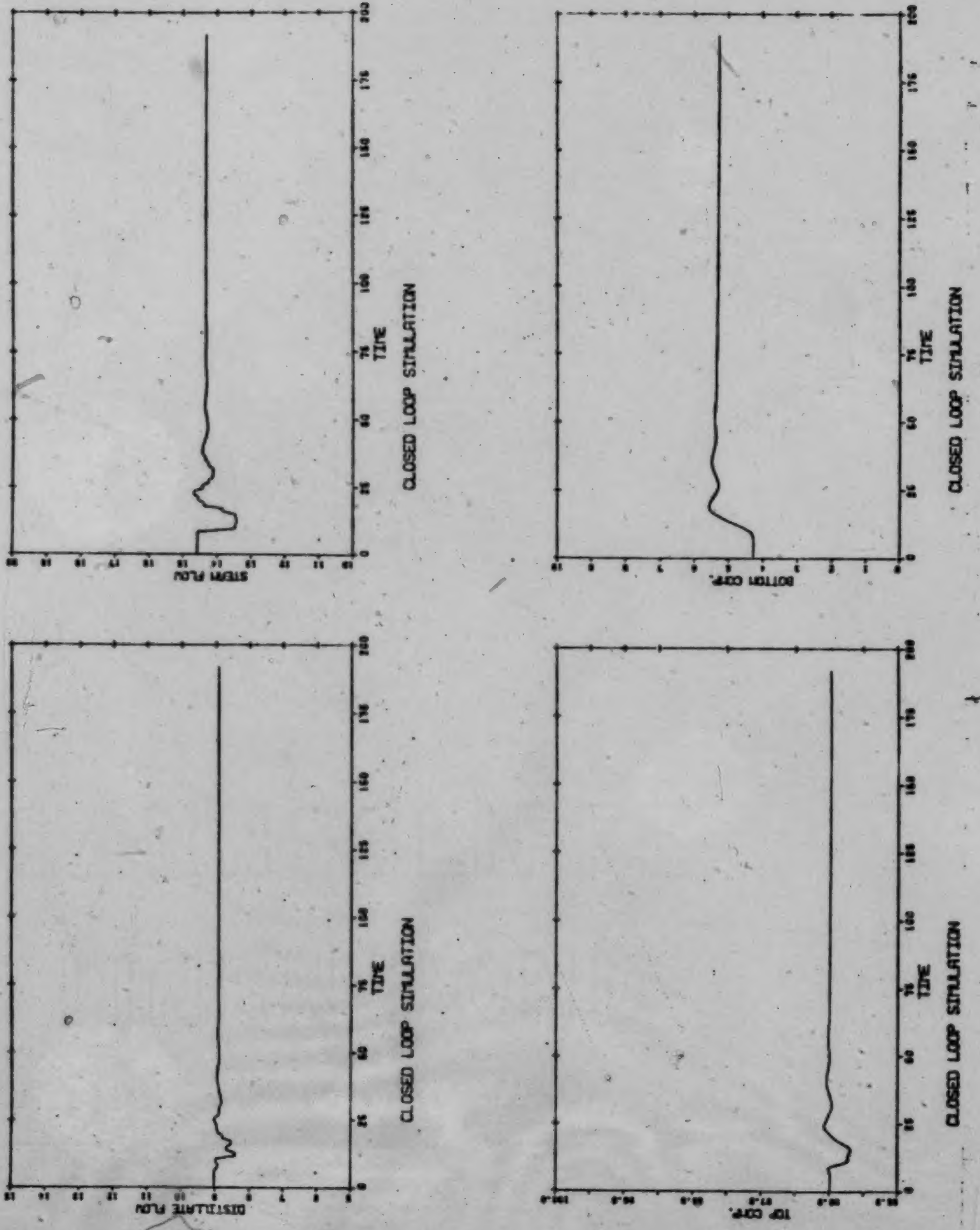
Plot 14. PID-plus-feedforward control, -20% feed flow rate SIMULATION RUN (flows in grams/sec, compositions in weight fraction methanol)



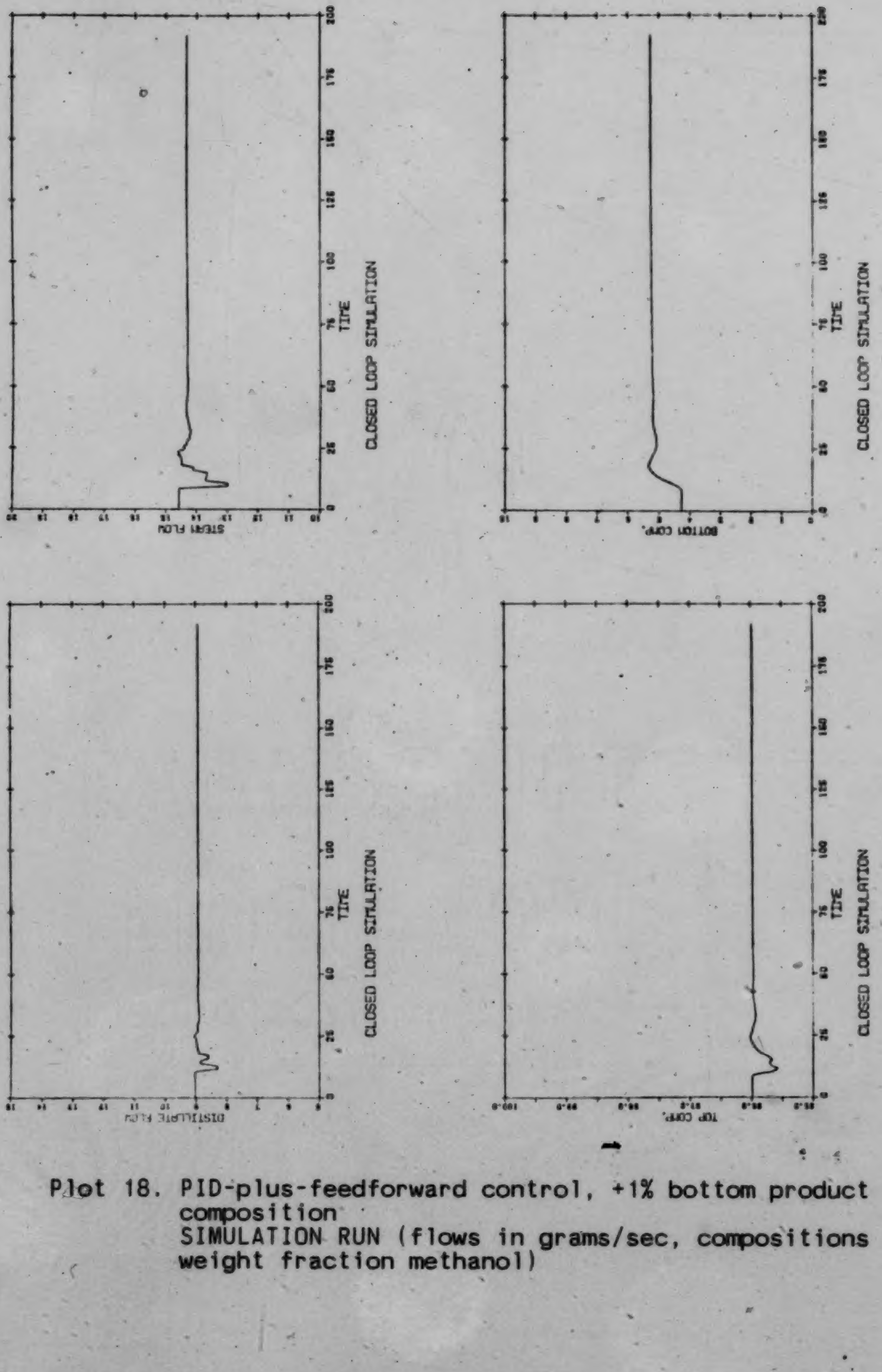
Plot 15. PD-robust feedback feedforward control, +1% top product composition
SIMULATION RUN (flows in grams/sec, compositions in weight fraction methanol)



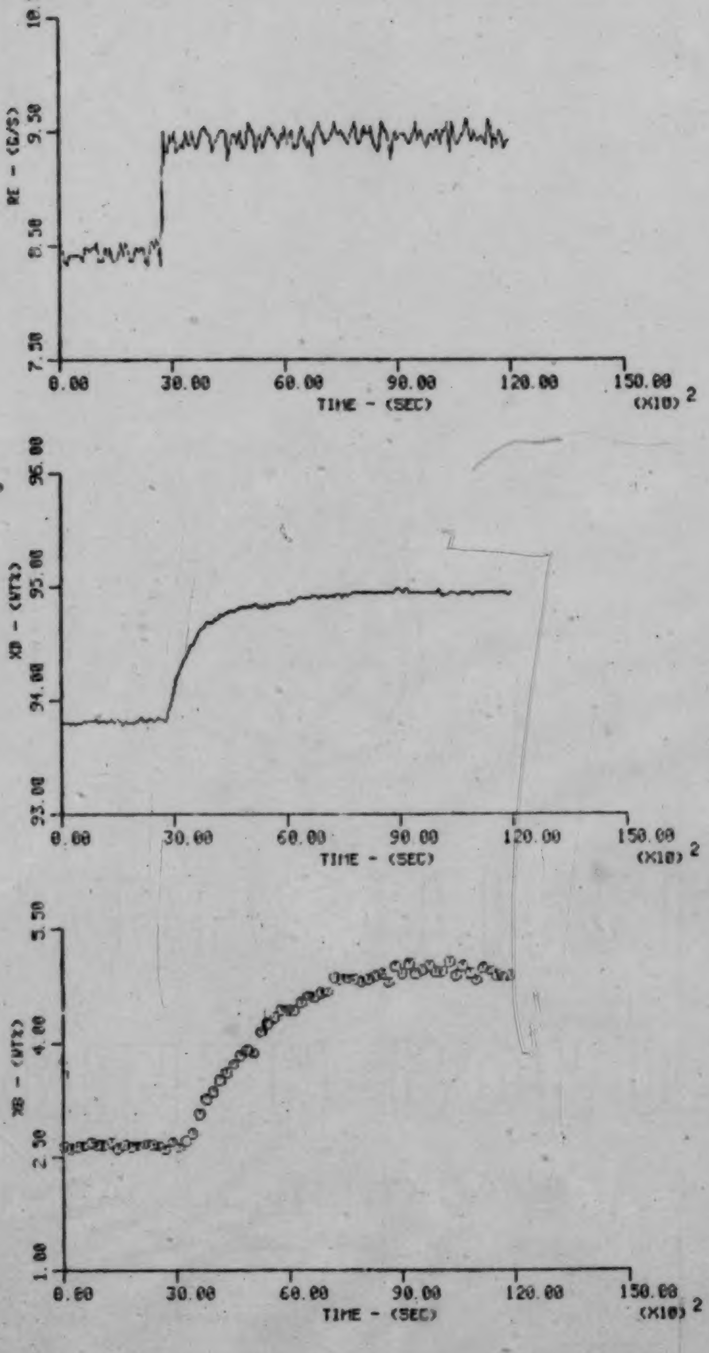
Plot 16. PID-plus-feedforward control, +1% top product composition
SIMULATION RUN (flows in grams/sec, compositions in weight fraction methanol)



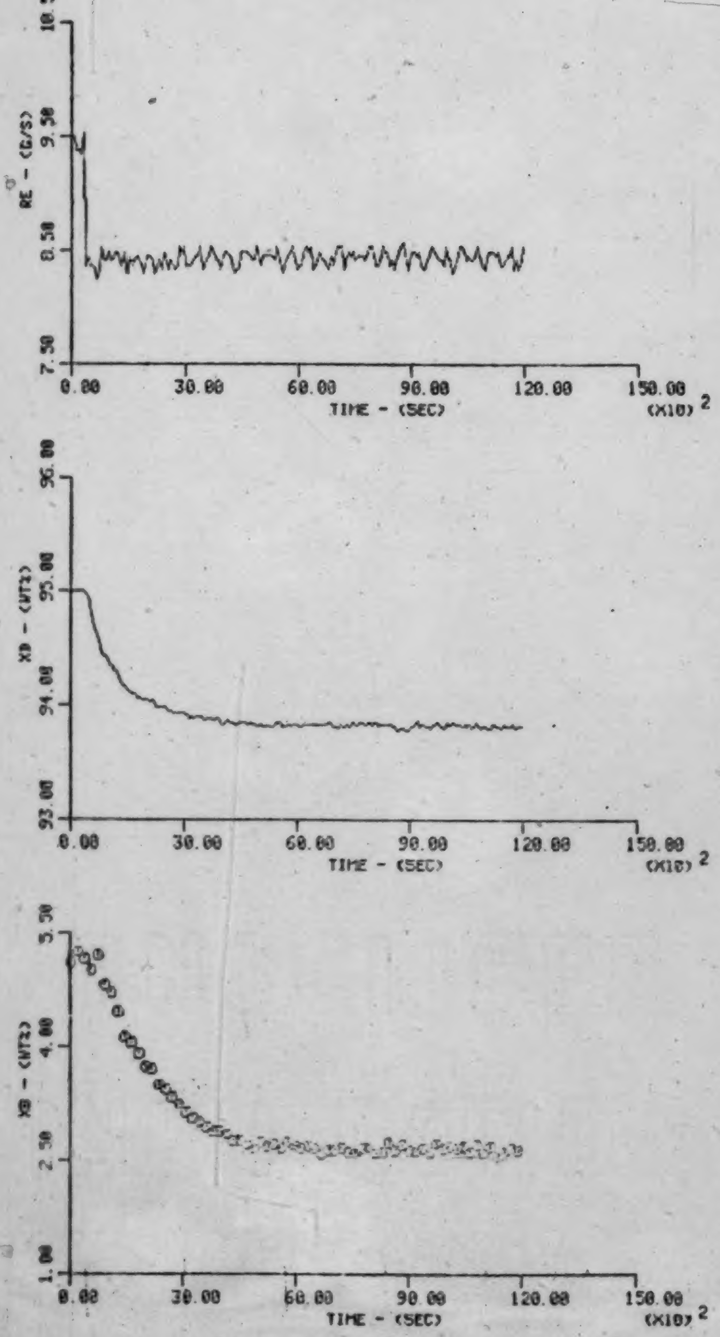
Plot 17. PD-robust feedback feedforward control, +1% bottom product composition
SIMULATION RUN (flows in grams/sec, compositions in weight fraction methanol)



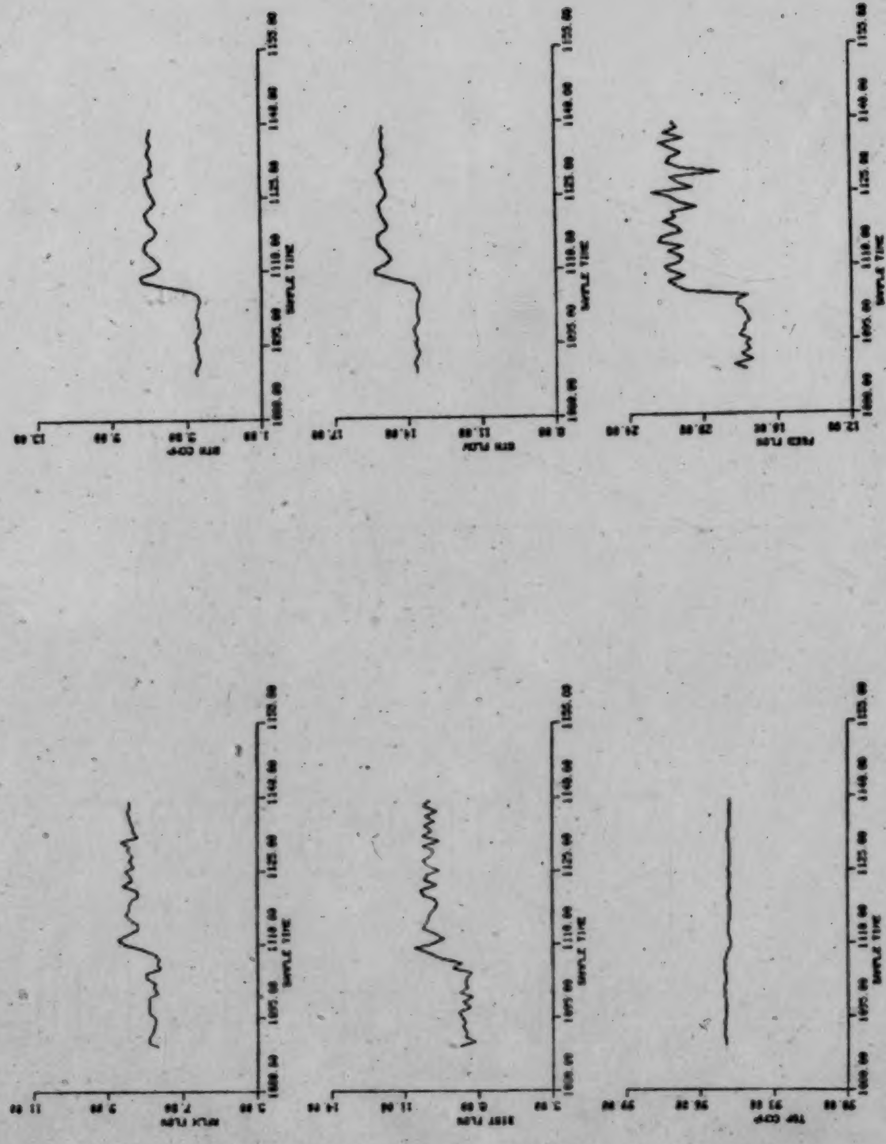
Plot 18. PID-plus-feedforward control, +1% bottom product composition
SIMULATION RUN (flows in grams/sec, compositions in weight fraction methanol)



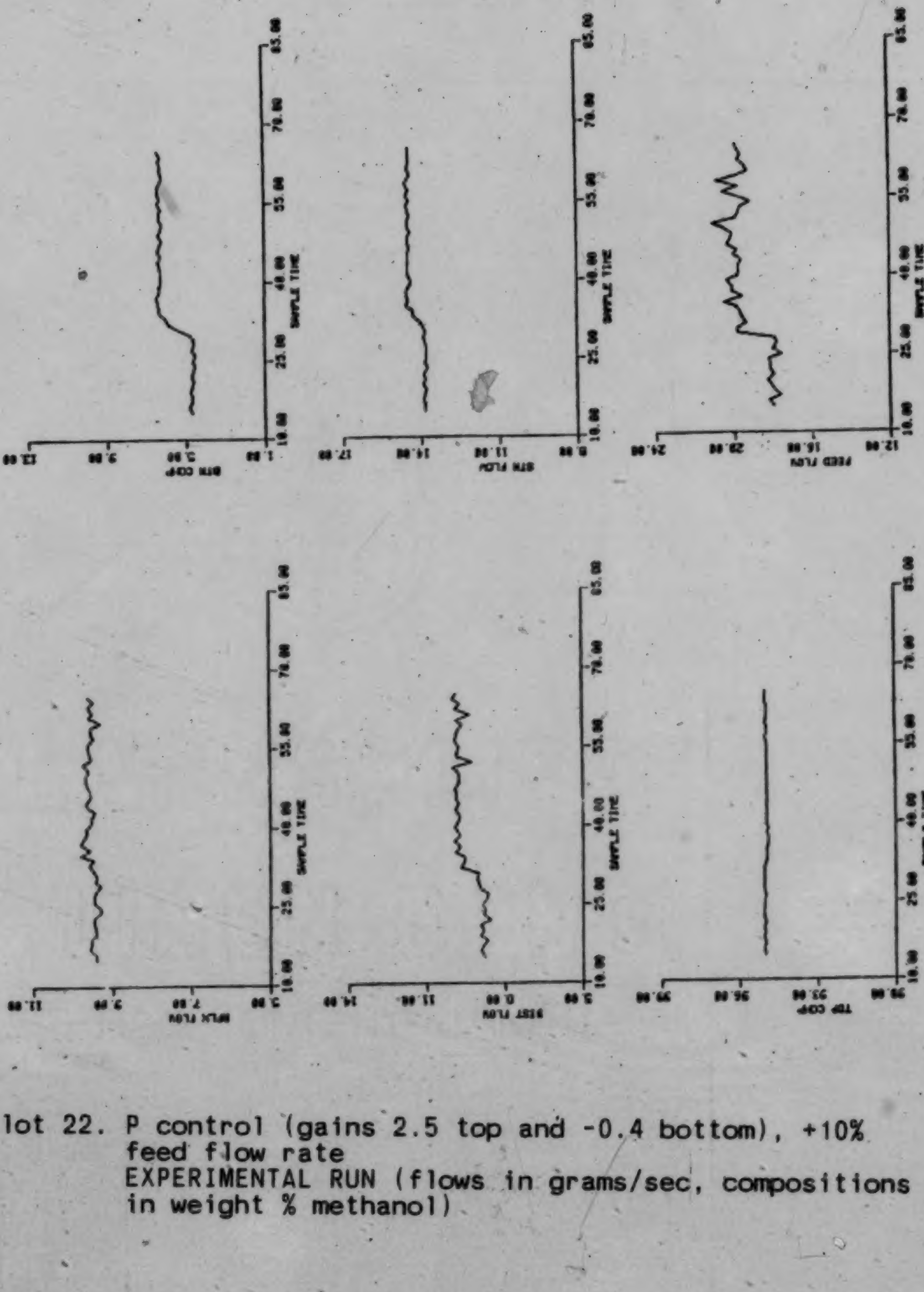
Plot 19. Open loop, +10% reflux flow rate
EXPERIMENTAL RUN (flows in grams/sec, compositions in weight % methanol)



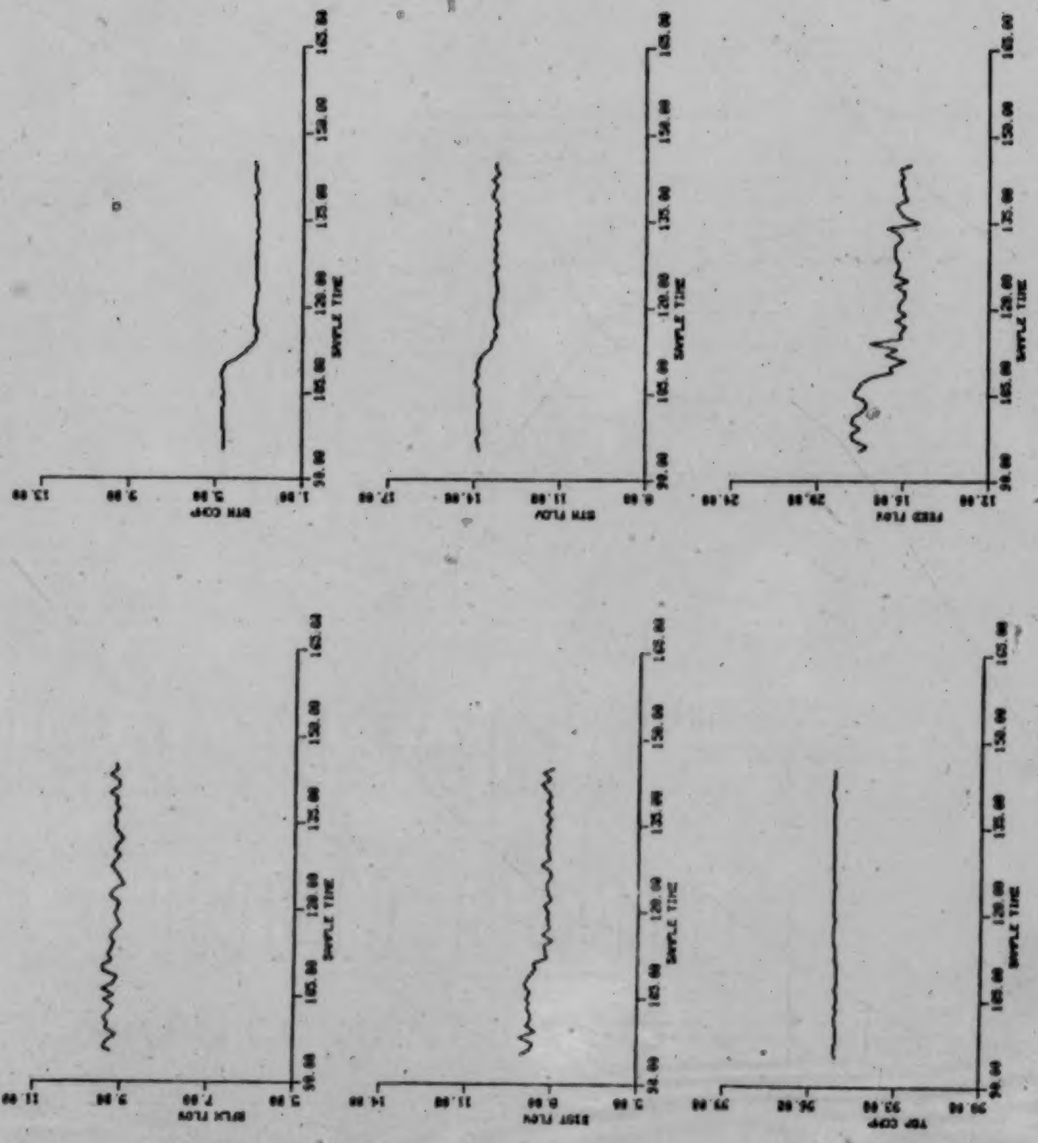
Plot 20. Open loop, -10% reflux flow rate
EXPERIMENTAL RUN (flows in grams/sec, compositions in weight % methanol)



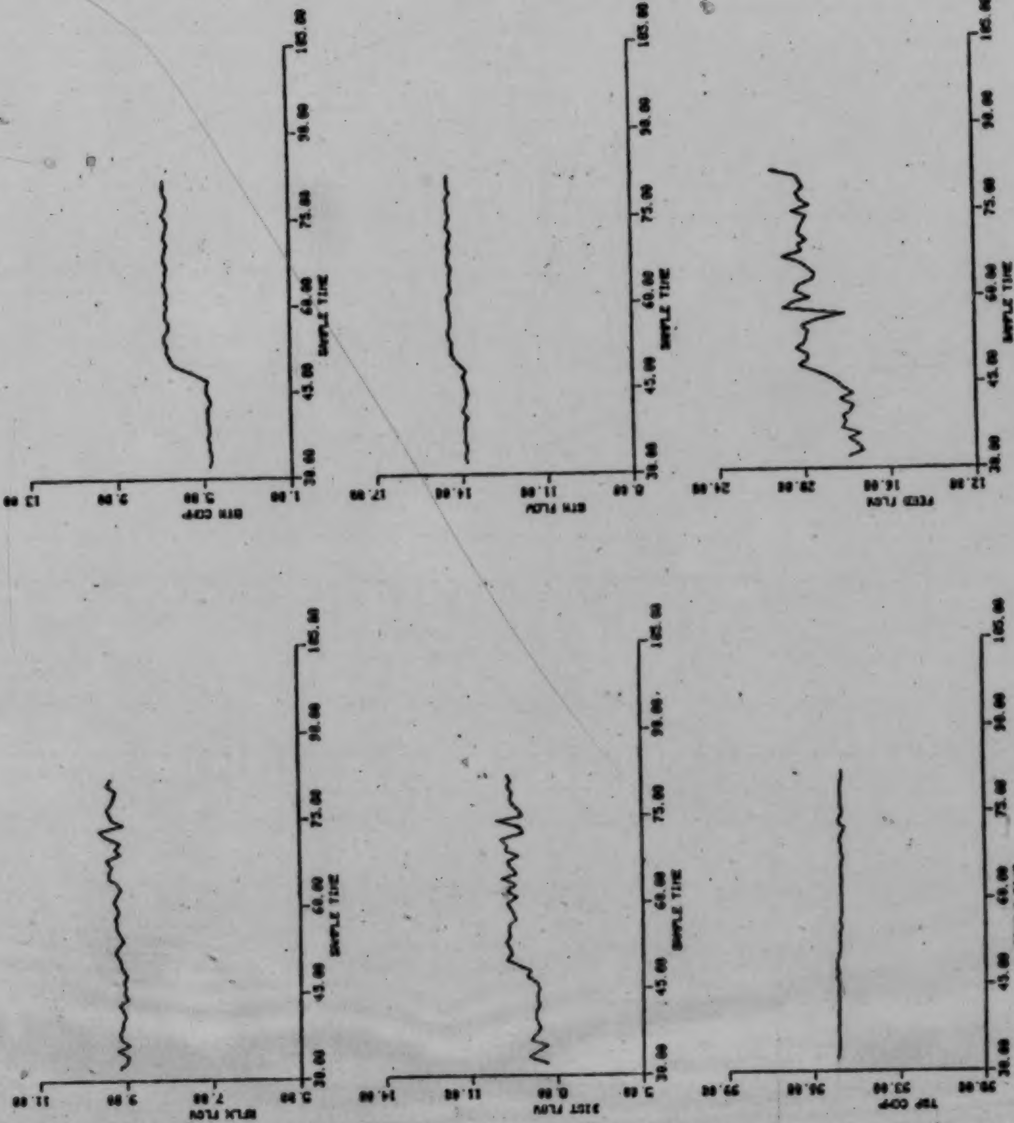
Plot 21. P control (gains 3.571 top and -0.576 bottom), +20% feed flow rate
EXPERIMENTAL RUN (flows in grams/sec, compositions in weight % methanol)



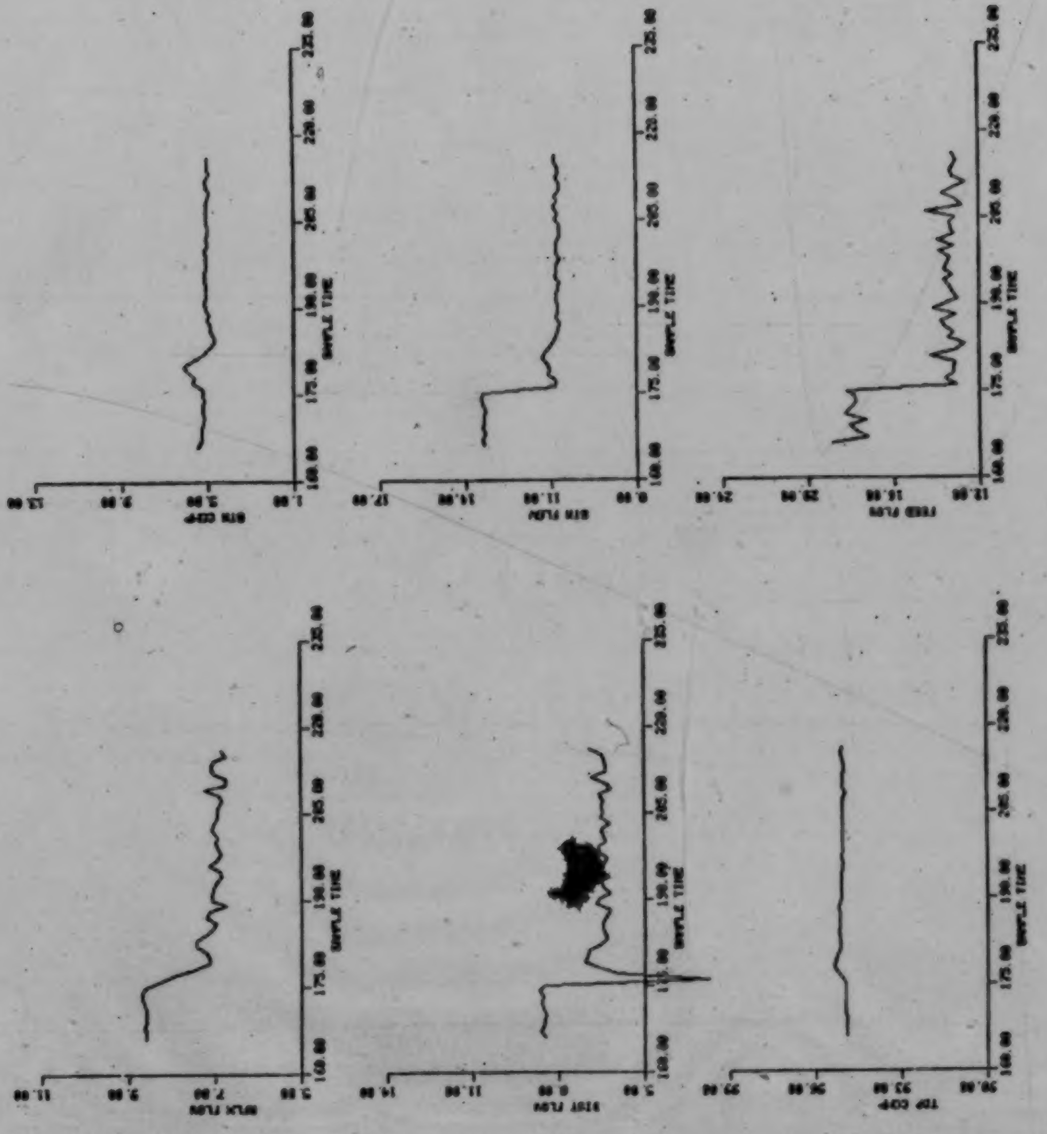
Plot 22. P control (gains 2.5 top and -0.4 bottom), +10% feed flow rate
EXPERIMENTAL RUN (flows in grams/sec, compositions in weight % methanol)



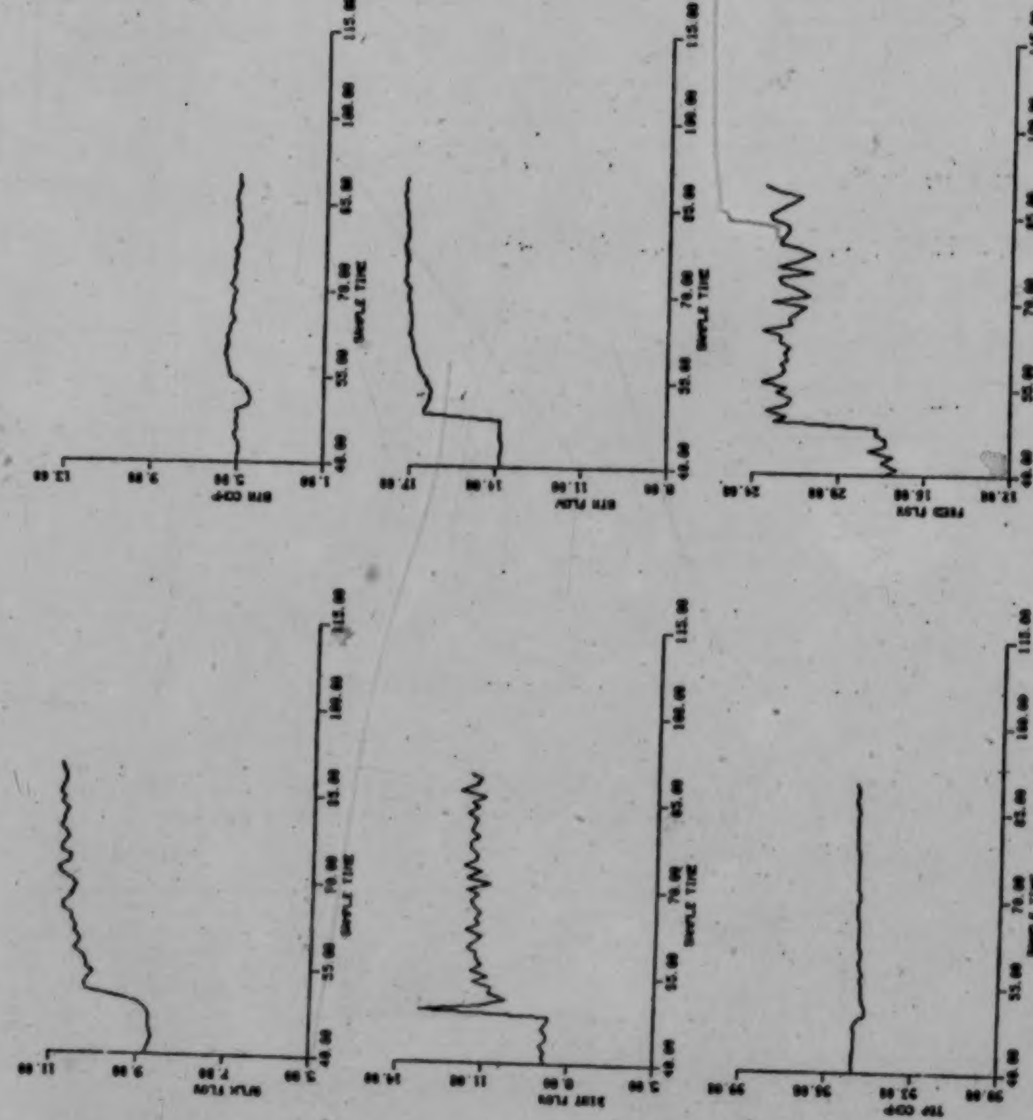
Plot 23. P control (gains 2.5 top and -0.4 bottom), -10% feed flow rate
EXPERIMENTAL RUN (flows in grams/sec, compositions in weight % methanol)



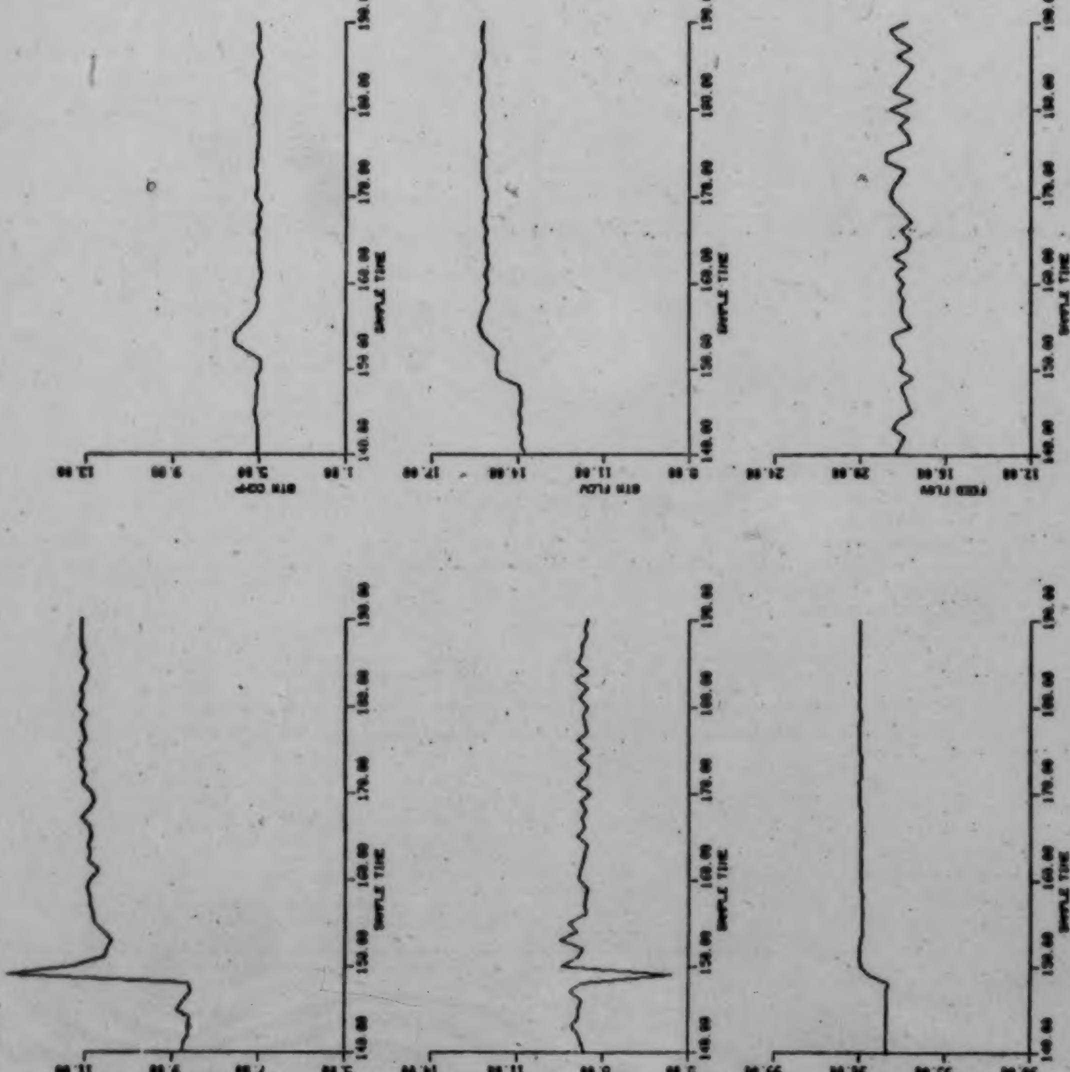
Plot 24. P control (gains 1.786 top and -0.288 bottom), +10% feed flow rate
EXPERIMENTAL RUN (flows in grams/sec, compositions in weight % methanol)



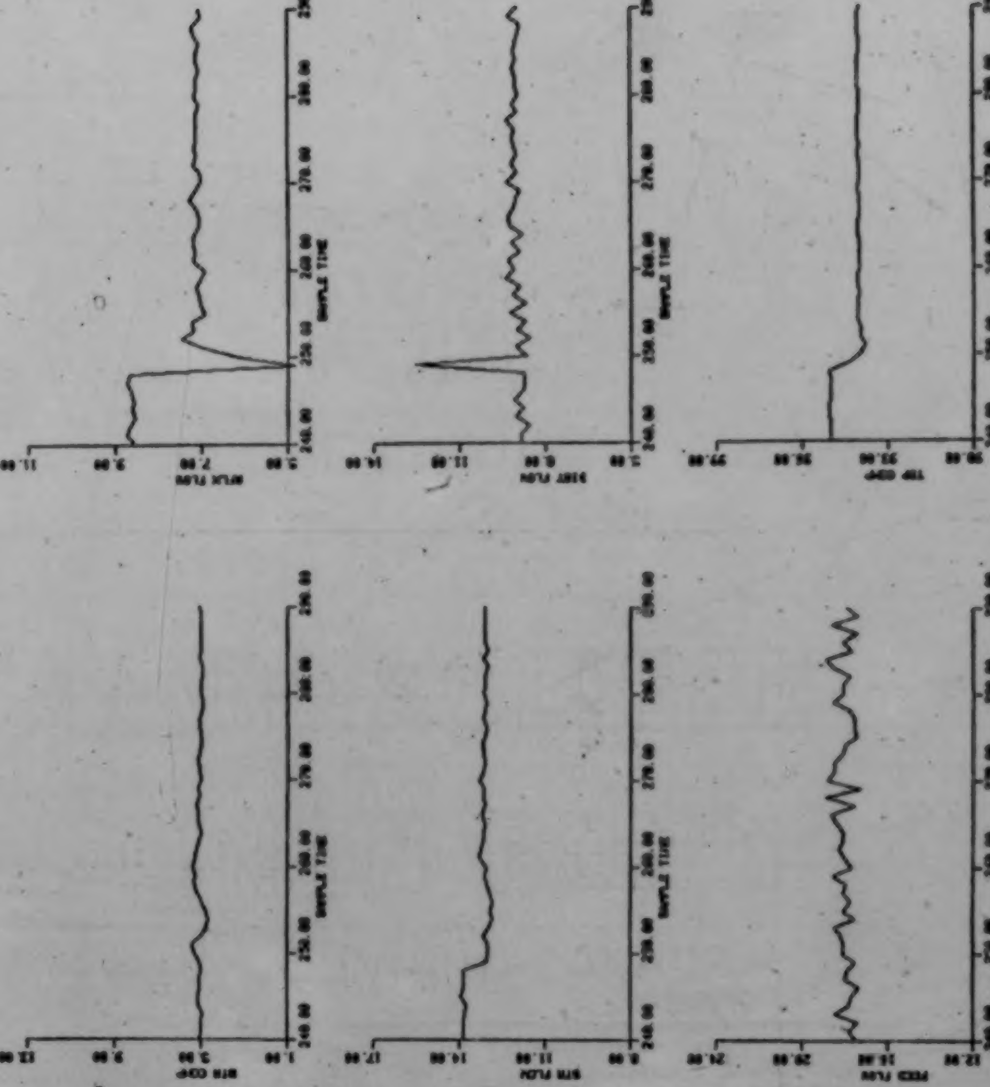
Plot 25. P-robust feedback feedforward control, -25% feed flow rate
EXPERIMENTAL RUN (flows in grams/sec, compositions in weight % methanol)



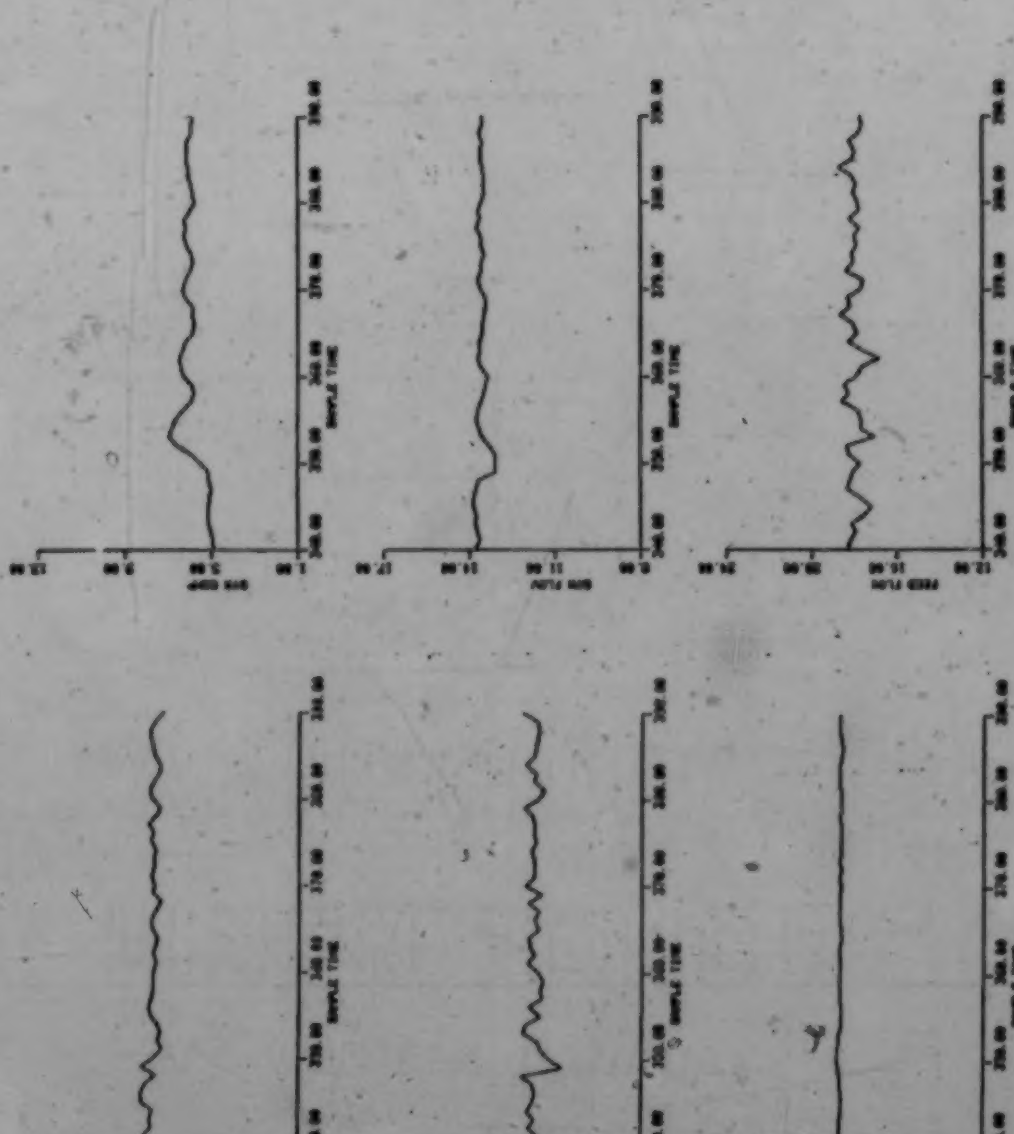
Plot 26. P-robust feedback feedforward control, +25% feed flow rate
EXPERIMENTAL RUN (flows in grams/sec, compositions in weight % methanol)



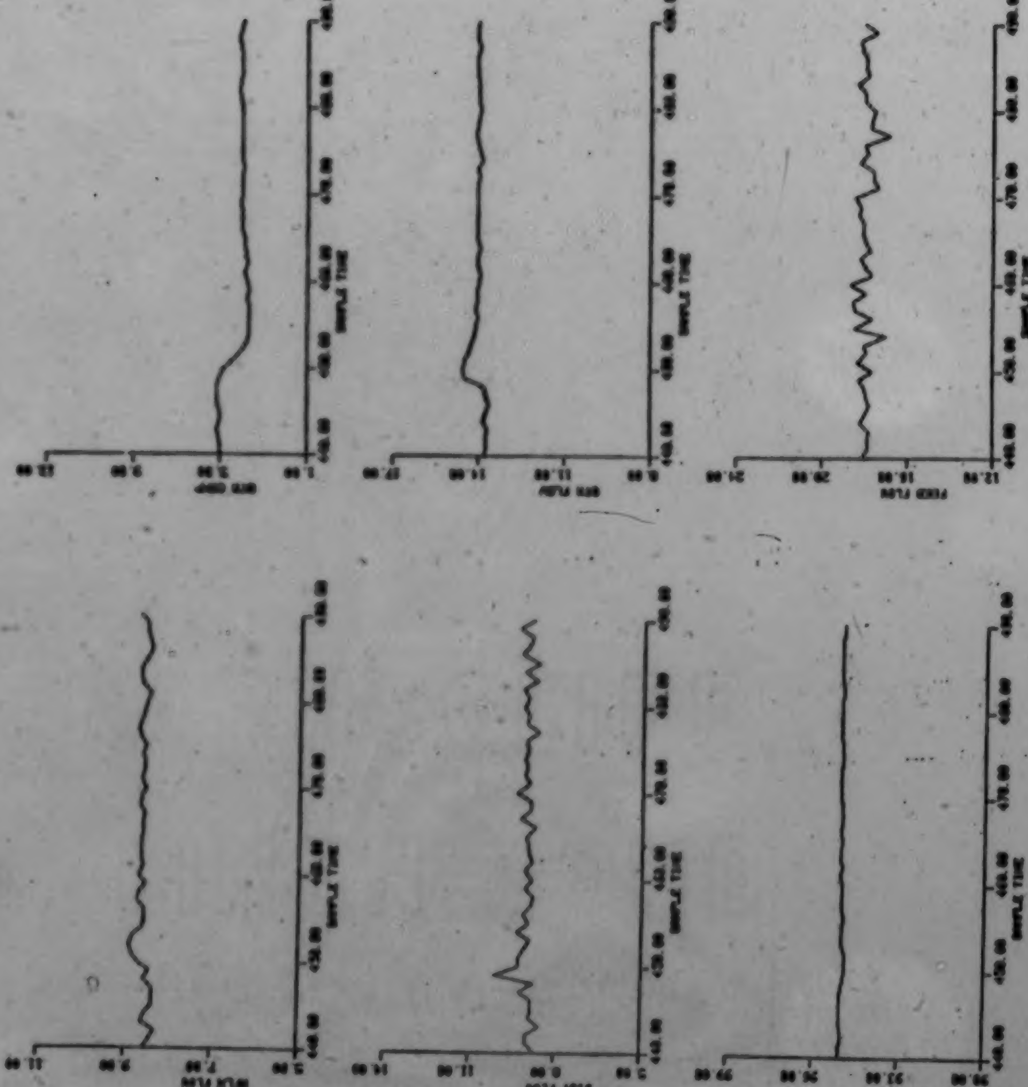
Plot 27. P-robust feedback feedforward control, +1% top product composition
EXPERIMENTAL RUN (flows in grams/sec, compositions in weight % methanol)



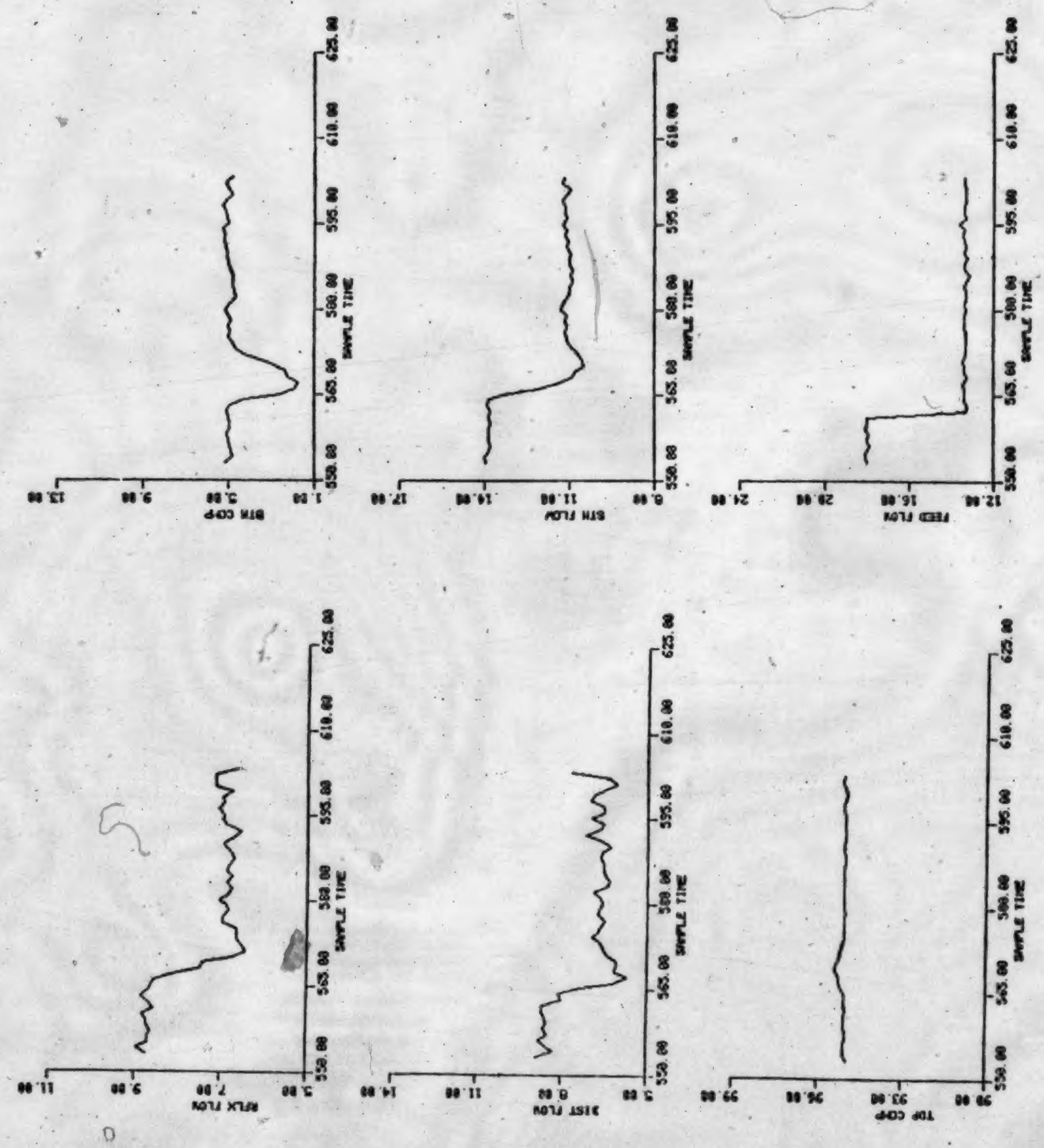
Plot 28. P-robust feedback feedforward control, -1% top product composition
EXPERIMENTAL RUN (flows in grams/sec, compositions in weight % methanol)



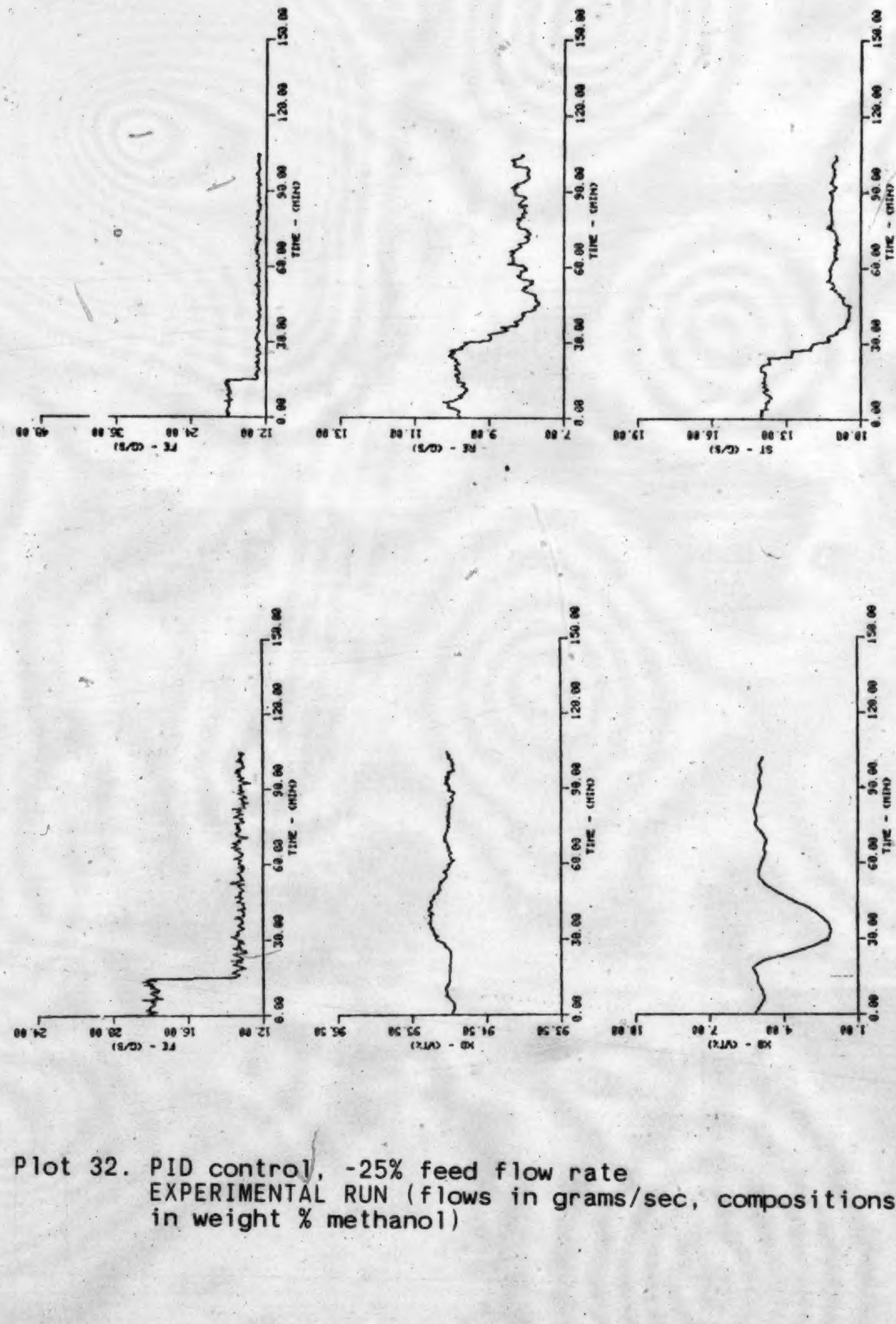
Plot 29. P-robust feedback feedforward control, +1% bottom product composition
EXPERIMENTAL RUN (flows in grams/sec, compositions in weight % methanol)



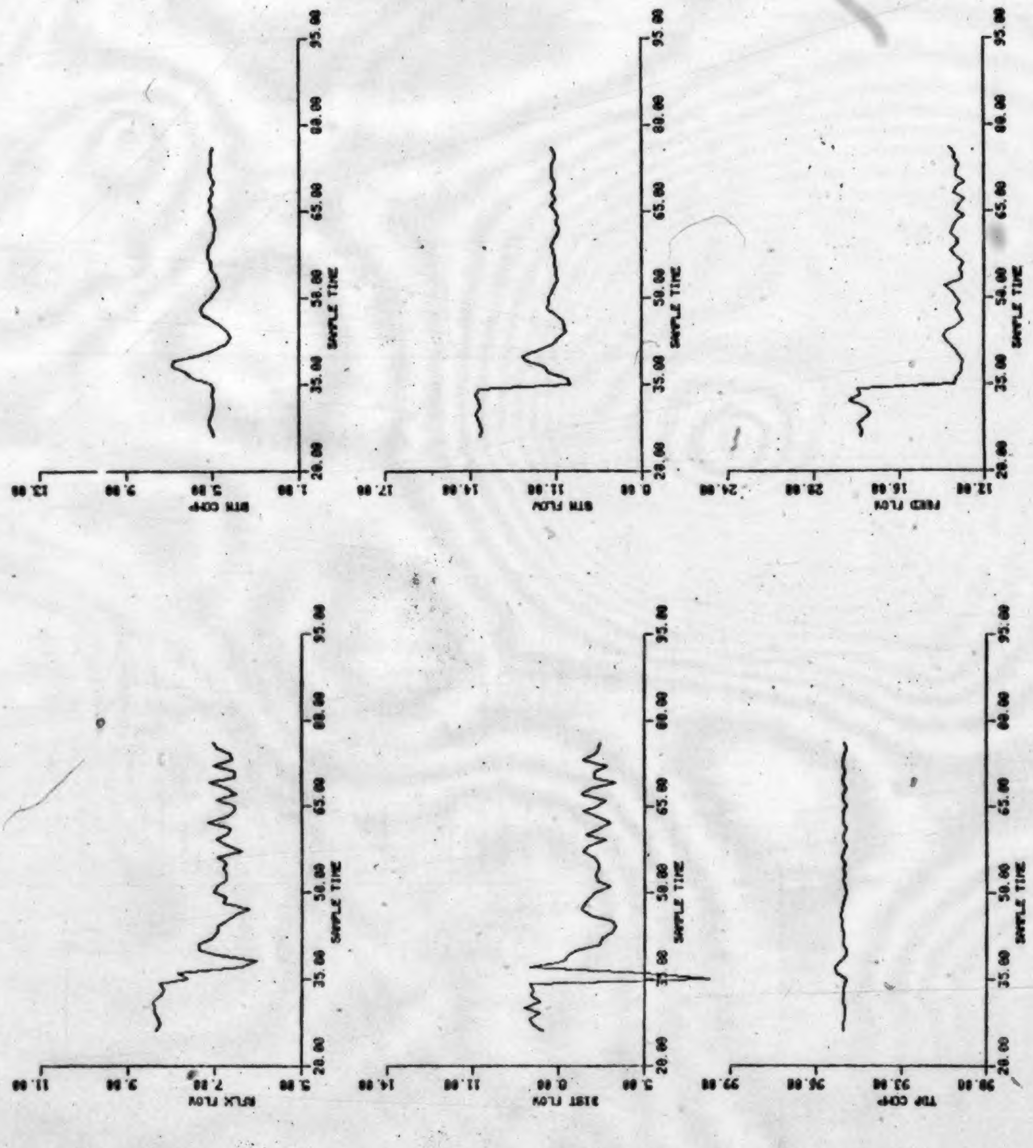
Plot 30. P-robust feedback feedforward control, -1% bottom product composition
EXPERIMENTAL RUN (flows in grams/sec, compositions in weight % methanol)



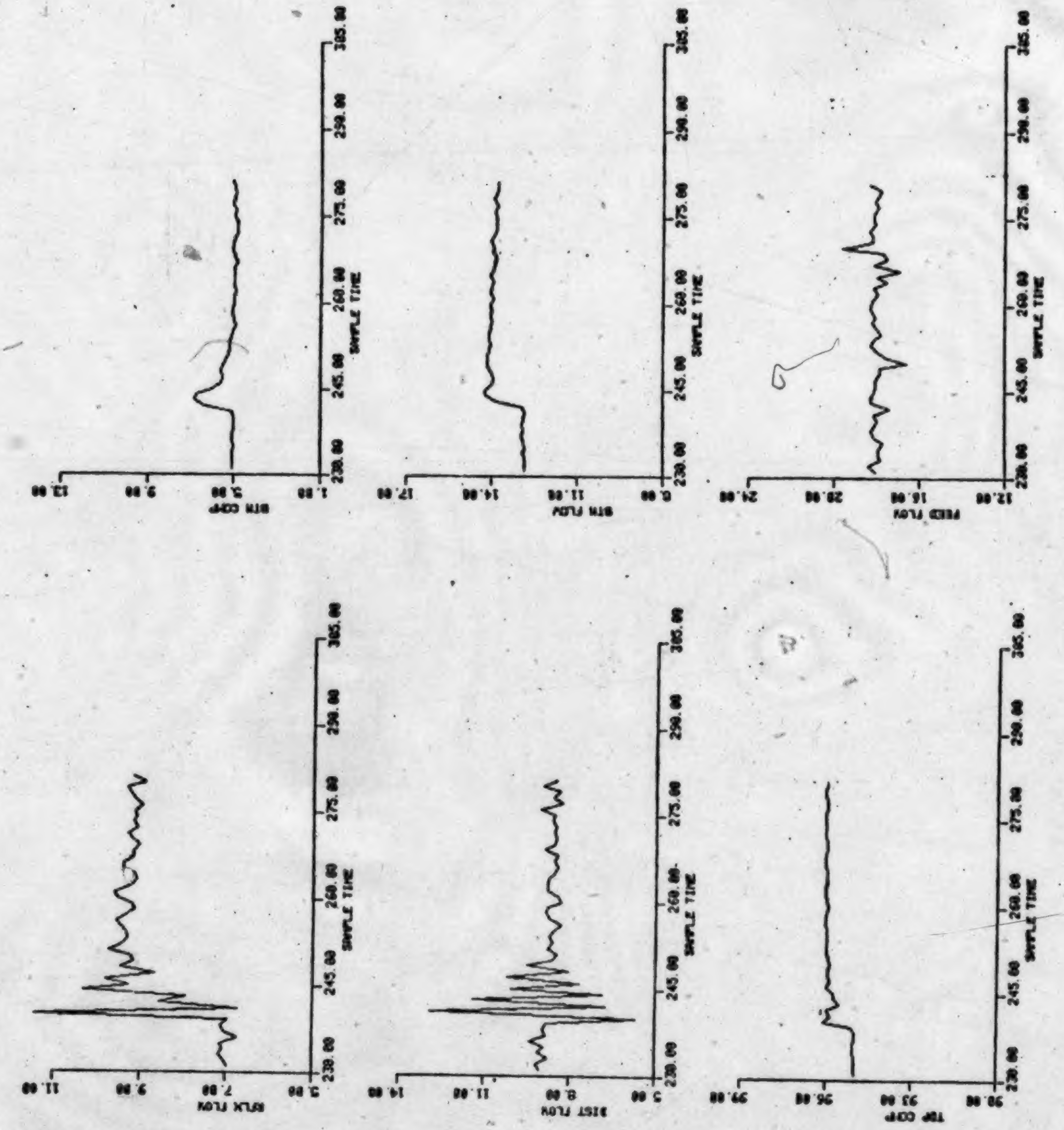
Plot 31. PI control, -25% feed flow rate
EXPERIMENTAL RUN (flows in grams/sec, compositions
in weight % methanol)



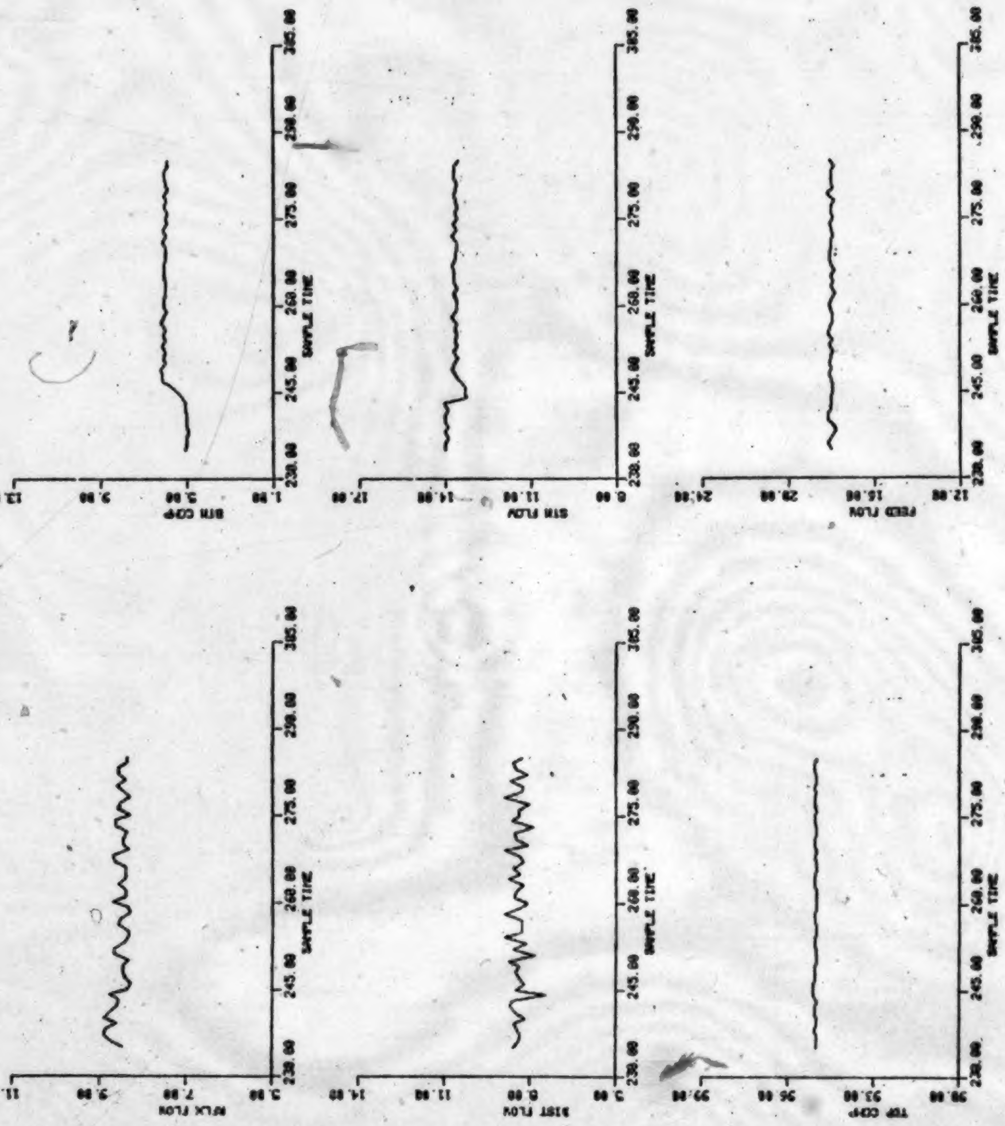
Plot 32. PID control, -25% feed flow rate
EXPERIMENTAL RUN (flows in grams/sec, compositions in weight % methanol)



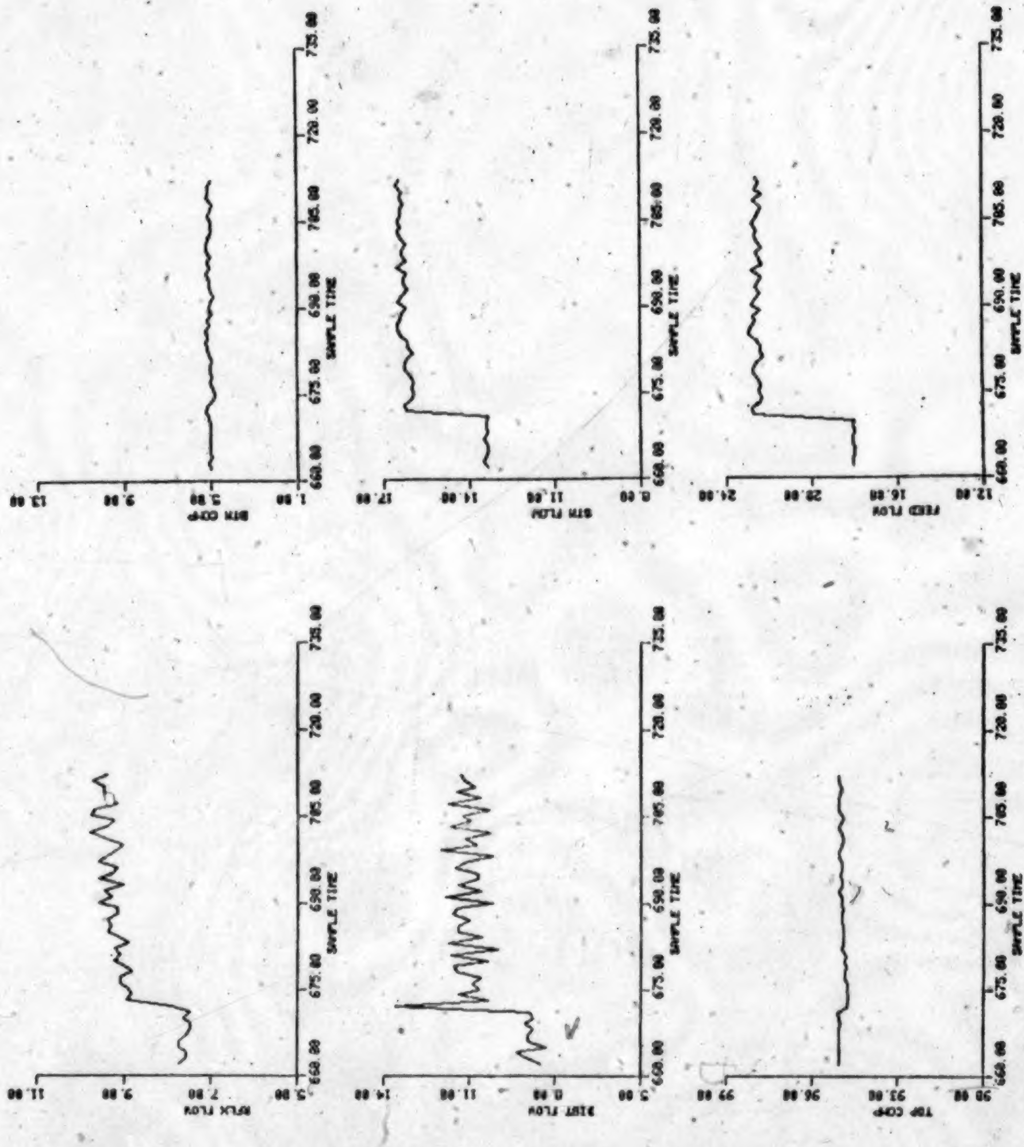
Plot 33. PID-plus-feedforward control, -25% feed flow rate EXPERIMENTAL RUN (flows in grams/sec, compositions in weight % methanol)



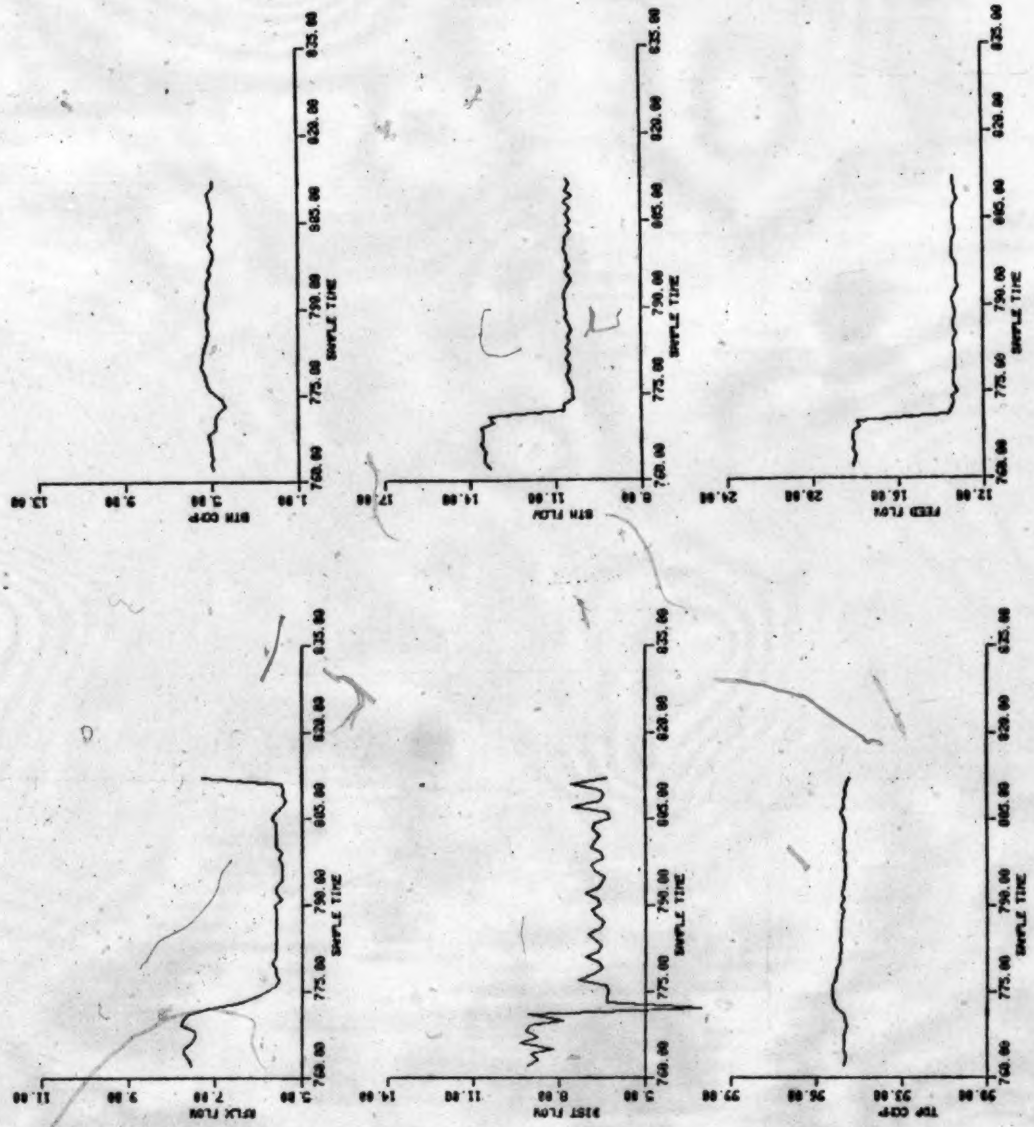
Plot 34. PID-plus-feedforward control, +1% top product composition
EXPERIMENTAL RUN (flows in grams/sec, compositions in weight % methanol)



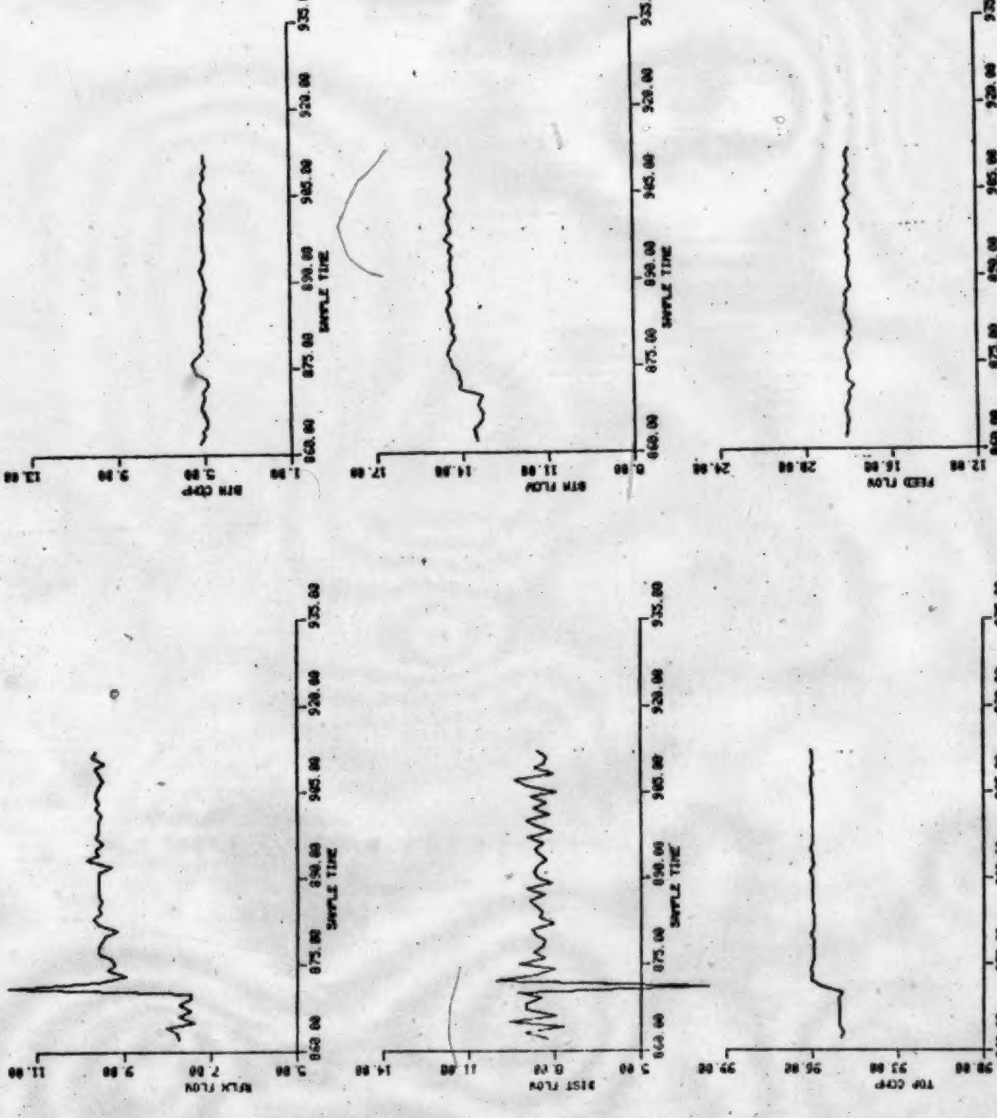
Plot 35. PID-plus-feedforward control, +1% bottom product composition
EXPERIMENTAL RUN (flows in grams/sec, compositions in weight % methanol)



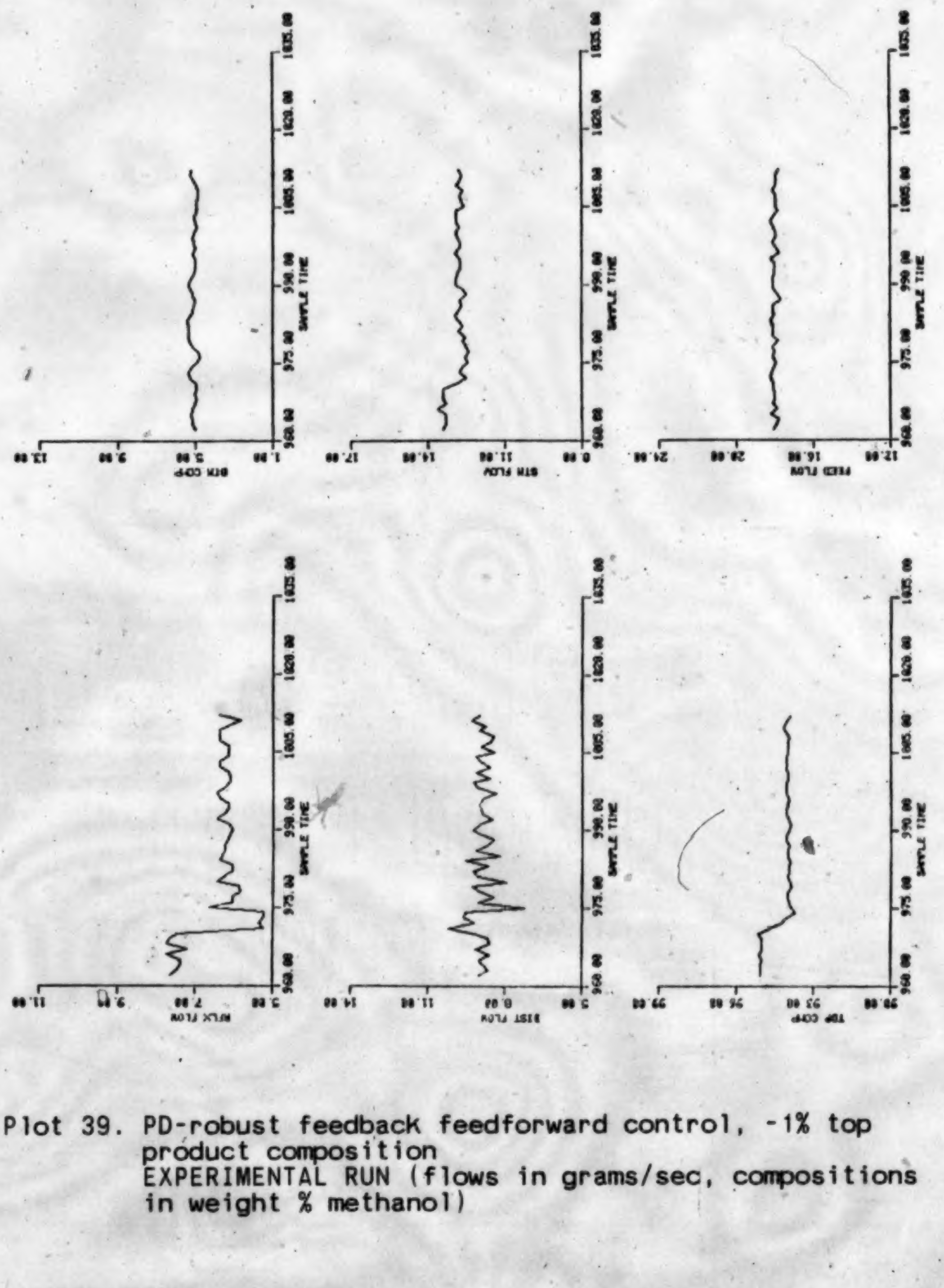
Plot 36. PD-robust feedback feedforward control, +25% feed flow rate
EXPERIMENTAL RUN (flows in grams/sec, compositions in weight % methanol)



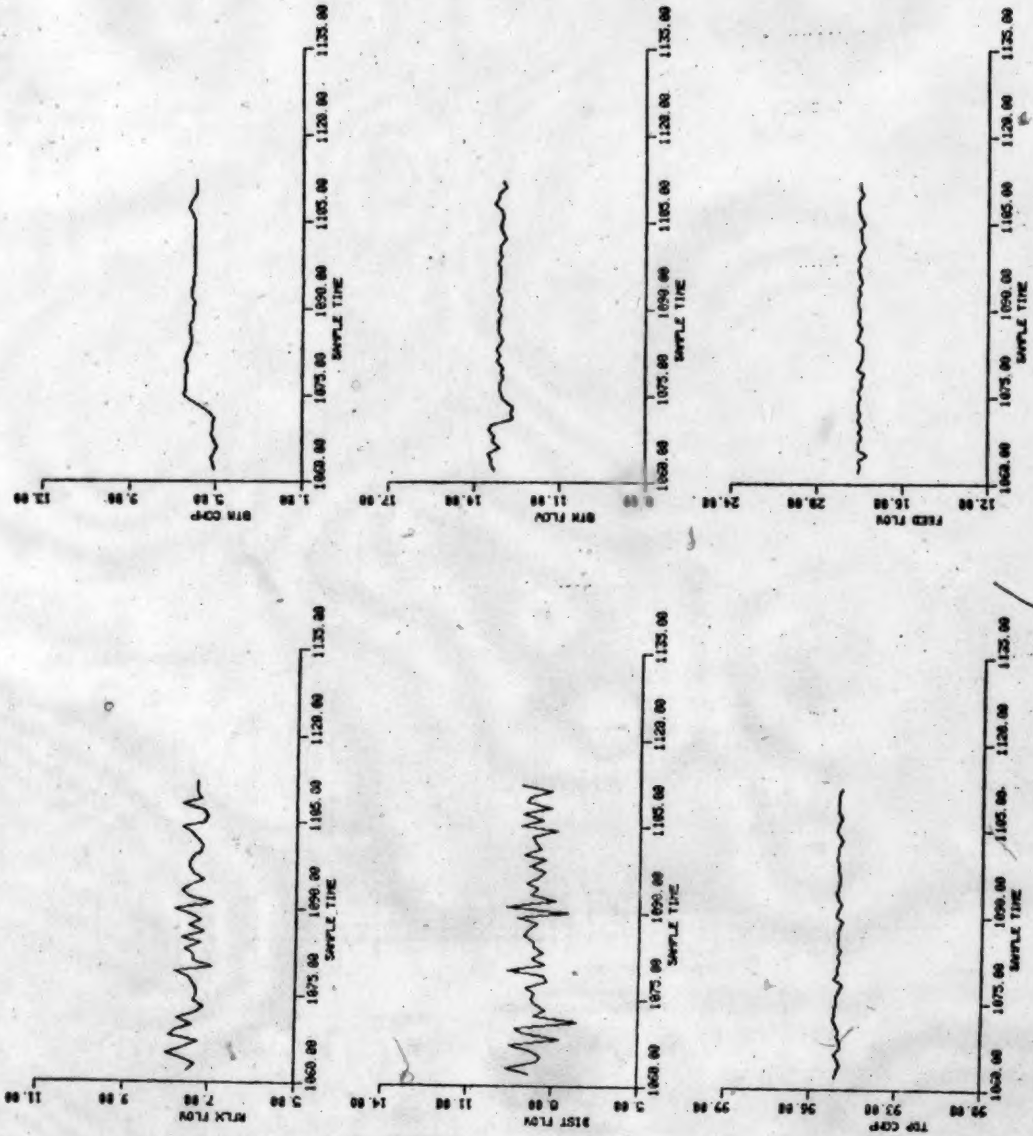
Plot 37. PD-robust feedback feedforward control, -25% feed flow rate
EXPERIMENTAL RUN (flows in grams/sec, compositions in weight % methanol)



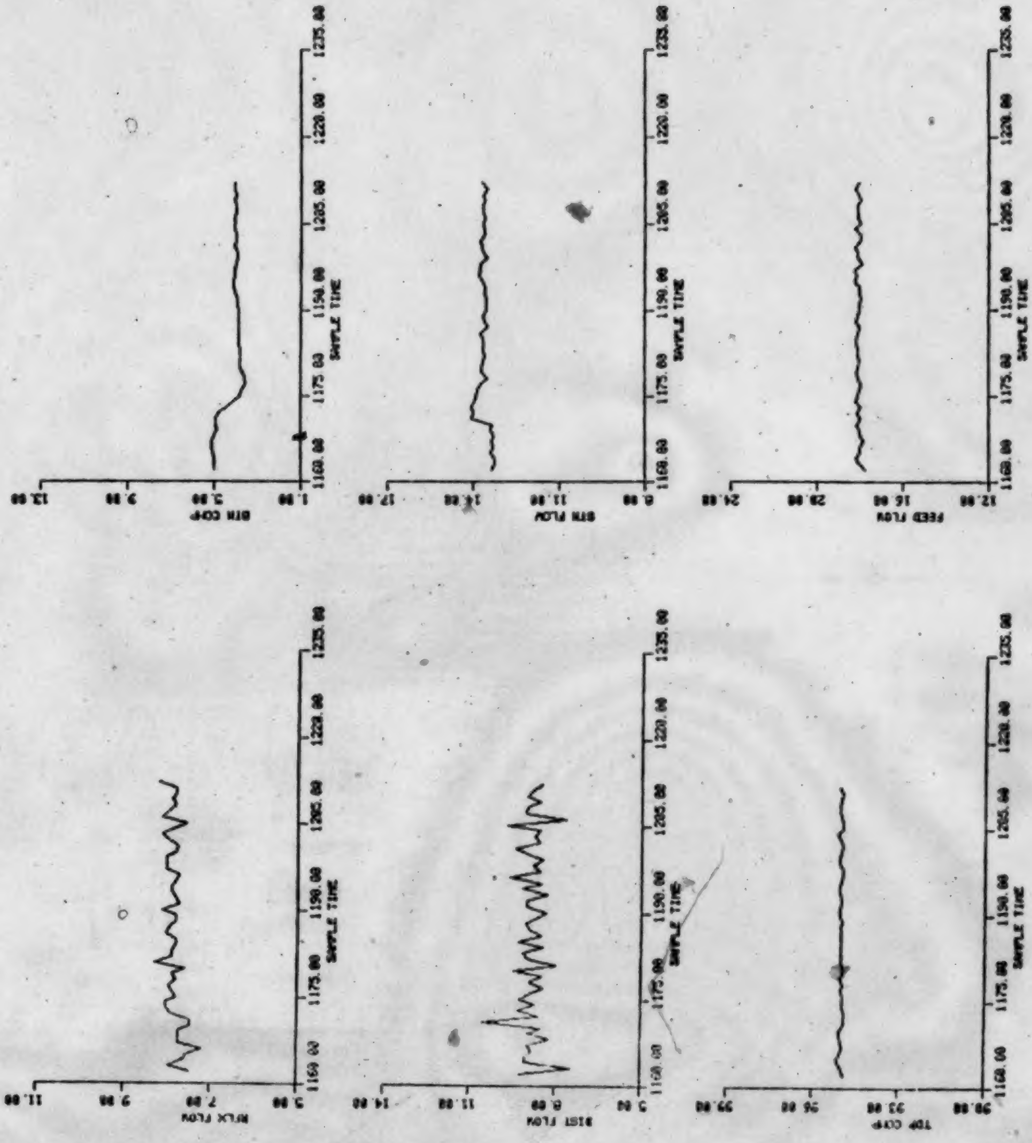
Plot 38. PD-robust feedback feedforward control, +1% top product composition
EXPERIMENTAL RUN (flows in grams/sec, compositions in weight % methanol)



Plot 39. PD-robust feedback feedforward control, -1% top product composition
EXPERIMENTAL RUN (flows in grams/sec, compositions in weight % methanol)



Plot 40. PD-robust feedback feedforward control, +1% bottom product composition
EXPERIMENTAL RUN (flows in grams/sec, compositions in weight % methanol)



Plot 41. PD-robust feedback feedforward control, -1% bottom product composition
EXPERIMENTAL RUN (flows in grams/sec, compositions in weight % methanol)